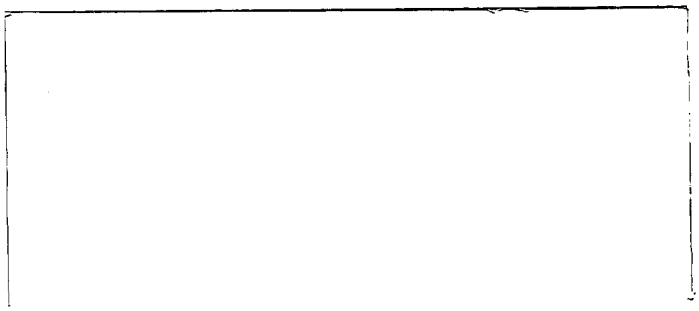
13446

COASTAL ZONE
INFORMATION CENTER



Texas Coastal Management Program

Texas. General Land Office

HT
391
.T49
1978

Texas Coastal Management Program
Technical Paper No. 14

COMPUTER ASSISTANCE

ACTIVITY ASSESSMENT ROUTINE
ECOLOGICAL SYSTEMS COMPONENT

Property of CSC Library

The General Land Office of Texas
Bob Armstrong, Commissioner

RPC, Inc.
Austin, Texas

August 1978

HT391.T49 1978
SEP 3 0 1997
U.S. DEPARTMENT OF COMMERCE NOAA
COASTAL SERVICES CENTER
2234 SOUTH HOBSON AVENUE
CHARLESTON, SC 29405-2413

This is one of a series of technical papers, which cover a variety of topics. For information concerning other technical papers in this series, or to order more copies of this paper, contact:

Elizabeth Christian Wilds
RPC, Inc.
1705 Guadalupe
Austin, Texas 78701

This paper was prepared under contract with the General Land Office of Texas, Coastal Management Program. The report was partially funded throug financial assistance provided by the Coastal Zone Management Act of 1972, administered by the Office of Coastal Zone Management, U.S. Department of Commerce.

FOREWORD

This technical paper is one of a series of papers in which the background material, models, and data used to develop the ecological systems component (ESC) of the activity assessment routine (AAR) are discussed. Together, the papers are reference sources for the ESC user's manual and form a basis for further system development.

Staff members of the Environmental Management Division, Texas General Land Office, in Austin are available to assist interested parties in learning to use the system, and they welcome any questions, comments, and suggestions concerning the ESC.

Many individuals assisted in the production of these technical papers. The principal-in-charge was Ron Luke. Project manager was Andrew Reed. Author of this paper was William Brogden. James Kimmel managed final production of these technical papers. The technical editor was Nancy Grona. Production assistance was provided by Lori Snyder, Kim Frazier, Alice Adams, and Ray Helmers.

Bob Armstrong

Bob Armstrong, Commissioner
General Land Office of Texas

TABLE OF CONTENTS

LIST OF FIGURES

1. OVERVIEW OF THE COMPUTER-ASSISTED ESC PROGRAM SYSTEM

This technical paper provides an introduction to and documentation of programs that have been written to provide computer assistance for the ecological systems component (ESC) of the activity assessment routine. The purpose of these programs is to perform most of the bookkeeping functions which tend to be very time-consuming in the manual method. The computer does not perform any judgment tasks; it just keeps track of the analyst's decisions and comments and automatically provides a correctly formatted "worksheet" output of the end of the analysis.

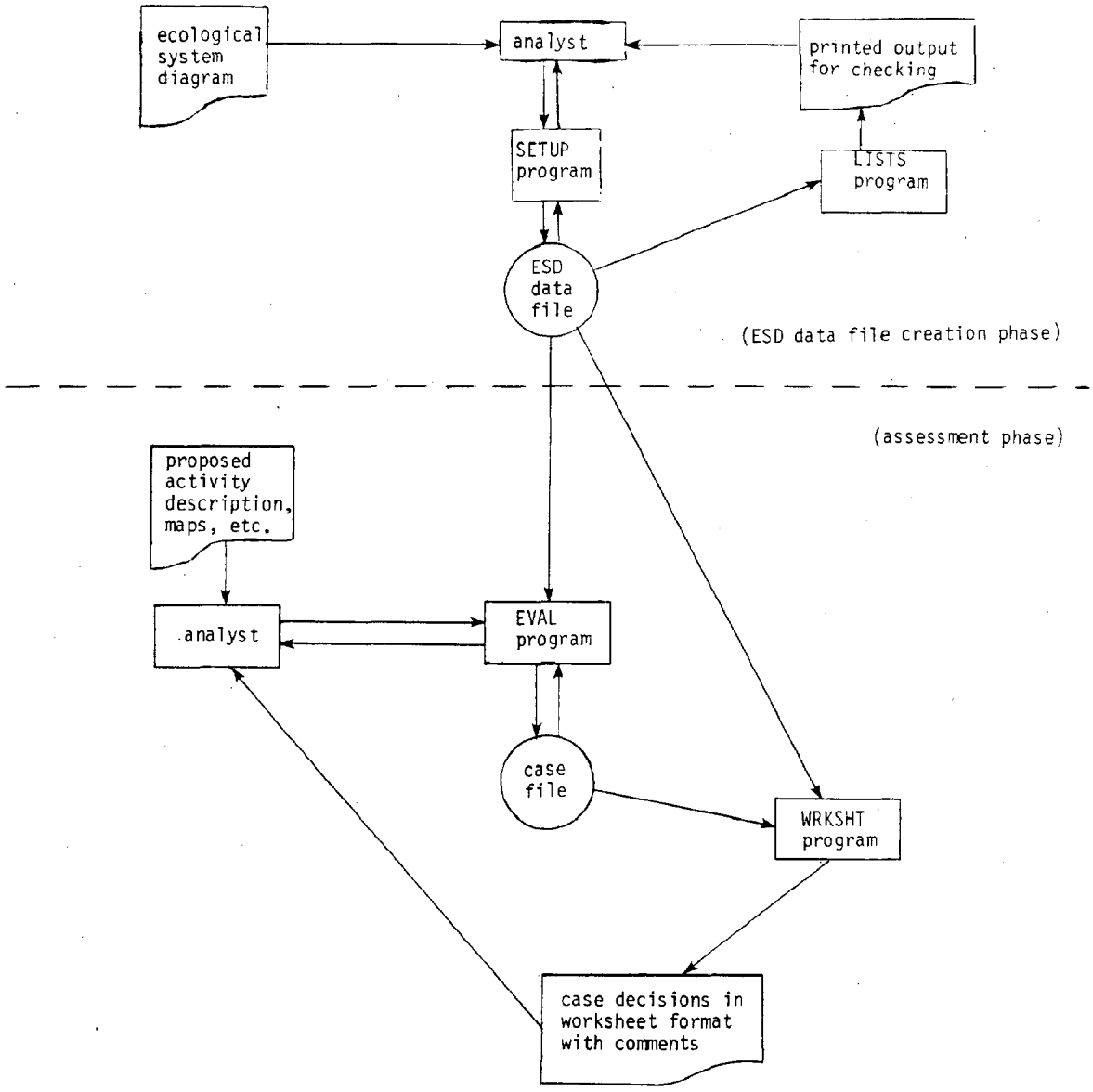
The potential user of these programs should be familiar with the manual method (described in the ESC user's manual) and with the operation of a computer terminal and timesharing system. The programs are presently implemented on the University of Texas at Austin timesharing system. Documentation for this timesharing system can be obtained at the Computation Center on the Austin campus.

Users of the program system will find all of the step-by-step instructions needed to operate the programs described in Chapter 2. Chapters 3 through 6 provide sufficient documentation for a programmer knowledgeable in FORTRAN to thoroughly understand the operation of the programs. These chapters include descriptions of data structure, program flowcharts, and the actual program listings.

The overall relationship between these programs and components of the ESC is depicted in Figure 1. An ecological system diagram (ESD) data base, which represents in a simplified way the interactions shown in an ecological system diagram, is created and edited with program SETUP which operates in an interactive mode. This data base also incorporates the primary ecological alterations (PEAs) which can occur in the ecosystem and their relationship to ecosystem attributes. Program LISTS is used to print out the relationships in a form which is easily used for error checking. Although the only data base which has been created so far is for the medium salinity bay ecosystem, it is a relatively simple process to create new ones, given an ecological system diagram.

Figure 1

RELATIONSHIP BETWEEN PROGRAMS, DATA FILES, AND OTHER COMPONENTS OF THE ESC



A proposed activity in a given ecological system can be evaluated using program EVAL exactly as described in the user's manual for manual methods, except that the program does almost all of the bookkeeping. Evaluation is conducted using an online interactive terminal, and all of the operator's decisions, comments, etc. are stored in a "case file." Analysis of a given activity can proceed in stages over many sessions with the computer, since the case file can be stored on tape and retrieved for subsequent sessions. At any stage in the analysis, the case file can be printed out in a format similar to the worksheet described in the user's manual by program WRKSHT.

LIMITATIONS OF COMPUTER ASSISTANCE

It is important to remember that the computer ESD data file does not contain all of the information incorporated in the ecological system diagram, just as the ESD is an abstraction of the available information on an ecological system. The computer does not make any judgments or check for consistency. The accuracy of the results will depend on the analyst and the available information sources. The advantages of the computer method are: (1) the analyst's time is not consumed with bookkeeping, and (2) the program requires that all interactions implied by the ESD must be evaluated.

FUTURE EXTENSIONS OF COMPUTER ASSISTANCE

The naming and numbering convention for ESD attributes which is used in the ESD data file could form the structure for an information retrieval system to give the analyst rapid access to the information needed during evaluation.

2. STEP-BY-STEP PROCEDURE FOR RUNNING THE PROGRAMS

RUNNING THE SETUP PROGRAM

This program is run only when it is desired to create a new ESD data base or to modify an old one. In preparation, the analyst should study the ESD data base structure (Figure 4, Chapter 3) and should have prepared a number scheme for attributes on an ecological system diagram. The analyst should also have determined in advance which PEAs can occur in the system.

1. Establish terminal connection with TAURUS timesharing service. (Documentation available at UT computation center)

2. Get working copy of the program (and old data base if modification is required).

   Example: READPF(9290,BSETUP,MEDSAL)

3. Start the program using the following format:

   BSETUP(TTY,TTY,OUTPUT,dbname) where dbname is the name of the data base file

4. The program will ask OLD OR NEW? O/N, enter either O for old or N for new. (Remember that a carriage return is needed before TAURUS will pass input on to the program.)

5. If O is entered, the program will read the old data base, print the title and number of attributes presently known, and then display the names of the attributes.

6. If N is entered, the program will request a new data base name. The maximum length accepted is 70 characters.

7. The program will now accept new attribute names until either the maximum number of names (currently 50) is reached or a blank line is entered. If no new attribute names are to be entered, the operator just sends a blank line by pressing return. The order in which names are entered establishes the numbering system used for attributes throughout the entire program system, so be careful.

8. Next, the operator has a chance to replace previously entered names to correct spelling errors, etc. The number of the attribute must be entered. The program responds with the existing name and expects the new name to be entered. The only way to leave this section is to enter a zero when the program asks for an attribute number.

9. Next, the program enters the matrix modification mode. Entries in the matrix describe the effect an increase in the "first" attribute has on the "second."

   Possible entries are:

   1 = decrease
   2 = increase
   3 = relationship too complex to describe as 1 or 2
   0 = no effect - NOTE: all interactions are set to 0 when the matrix is created and do not have to be changed unless 1, 2, or 3 is desired.

   At the present time, only the fact that an interaction exists is used by the programs; however, future programs may use the type of interaction as a consistency check.

   After a "first" attribute number is entered, the name of that attribute is printed as a check. Next, the program expects the number of the "second" attribute. The name of the "second" attribute is also printed, and the program asks for the interaction type; enter 0, 1, 2, or 3. The program will continue asking for "second" attributes, keeping the "first" constant Thus all interactions of a single attribute can be input with a minimum of repetition. When you are finished with a "first" attribute, enter a 0 instead of a second attribute number. The program will then ask for a new "first" attribute. If all modifications have been completed or if it is desired to save the intermediate product, enter a 0 at this point.

10. The program now moves on to the PEAs. It asks if PEA modifications are desired. If the answer is yes, it first accepts new PEA names until a blank line is input. This format is analogous to (7) above.

11. After any new PEA names have been added, the operator is given a chance to alter existing PEA names by entering the number of a PEA to be changed. This format is analogous to (8) above.

12. Next, the program permits modification of the matrix describing which attributes can be altered by which PEA. The format is similar to that employed in (9) above, except that only 0 and 1 entries are used for no effect and possible effect.

13. Finally, the program offers three options:

    1 - Causes the data base to be printed on the OUTPUT file (named in (2) above). This will be printed at the computer center if the appropriate commands are given (see below).

    2 - Causes the modified data base to be written out as a local file. This is the only way to save the results of modifying or creating a data base.

    0 - This terminates the program. Options 1 and 2 return to the option selection program; option 0 is final.

14. In order to obtain the printed output produced by option 1, the user must execute the following commands after the SETUP program terminates:

    SPL(50) - set page limit to 50 decimal
    RSF. - release system files; in this case the file OUTPUT is released to the system printer

15. In order to retain a copy of the modified data base, it should be saved on a permanent file. See TAURUS documentation for details.

16. The printed output produced in (14) above is hard to interpret in terms of checking the data base for correctness of interaction representation. The LISTS program can be run at this time to give a more usable output format.

RUNNING THE LISTS PROGRAM

The purpose of this program is to print the interactions represented in an ESD data base in a form convenient for error checking. This is a batch type program; that is, it requires no operator input. However, since its most common use will be in conjunction with SETUP, the following discussion is given in terms of TAURUS operation.

1. Establish terminal connection with TAURUS.

2. Get working copies of the program and data base.

   Example: READPF(9290,BLISTS,dbname)
   Where dbname is the file name used for the data base.

3. Execute the program:

   BLISTS(dbname)

   The running time should be only a few seconds.

4. The printout is now on the system file named OUTPUT. Give the following commands to release it for printing:

   SPL(50) - set page limit
   RSF. - stands for release system files

5. The data base file is not altered by this program.

RUNNING THE EVAL PROGRAM

In order to use the EVAL program to assist in the evaluation of a proposed activity, an ESD data base for the affected ecosystem must be available. In addition, the analyst must have all of the project description maps and other information sources which the manual method requires. The following description assumes operation on the University of Texas TAURUS timesharing system. Detailed instructions for the use of TAURUS are available through the UT computation center.

1. Establish terminal connection with TAURUS.

2. Get working copies of the program, data base, (and case file, if you are resuming an old analysis).

   Example:

   READPF(9290,BEVAL,MEDSAL,CASE02)

3. Start the program, using the following format, where dbase is the name of the case file. If this is a new analysis, this command also creates the case file, so the analyst should choose the name carefully. File names must start with a letter and be seven characters or less in length.

   BEVAL(dbase,casen)

4. The system will respond "GO:" and the program will begin operation. The data base file is automatically read and the title is printed out. As shown in Figures 2 and 3, the analyst must respond NEW or OLD to the question "NEW OR OLD ANALYSIS?".

Figure 2

EVAL DIALOG WITH "NEW" ANALYSIS

```
GO:
 READ
 DATA BASE
MEDIUM SALINITY BAY SYSEM DIAGRAM MAY 1978, W. BROGDEN                JULY 6.3
 WITH   39 ATT, PEA=  13
NEW OR OLD ANALYSIS ?
NEW
 NEW ANALYSIS, ENTER TITLE
EXAMPLE CASE FOR TECHNICAL PAPER ON COMPUTER ASSISTED ESC
 TITLE
EXAMPLE CASE FOR TECHNICAL PAPER ON COMPUTER ASSISTED ESC
 OK? YES/NO ?
YES
 ENTER COMMENTS, TERMINATE WITH BLANK LINE
 ASSUME A SMALL DREDGING CASE IN MEDIUM SL ALINITY BAY

  13 POSSIBLE PEAS IN THIS ECOSYSTEM
DOES CONSUMER REMOVAL                          OCCUR? YES/NO
YES
DOES INORGANIC MATERIALS IN WATER              OCCUR? YES/NO
YES
DOES ORGANIC MATERIALS IN WATER                OCCUR? YES/NO
YES
DOES TOXIC MATERIALS IN WATER                  OCCUR? YES/NO
NO
DOES INORGANIC MATERIALS IN SEDIMENT           OCCUR? YES/NO
```


(continued)

(Figure 2, continued)

```
      4 INITIAL PLAS
    IS HERBIVORES AND DETRITIVORES
    SIGNIFICANTLY AFFECTED BY PEA CONSUMER REMOVAL
   NO
    IS BENTHIC COMMUNITY
    SIGNIFICANTLY AFFECTED BY PEA CONSUMER REMOVAL
   YES
    SHORT TERM EFFECT
(a) EVALUATE CHANGE IN BENTHIC COMMUNITY                    LEVEL  1 DUE TO
     CONSUMER REMOVAL                          = NOT EVAL.
    ENTER EFFECT ESTIMATE
   SH/M6/D/DF
    ENTER COMMENTS, TERMINATE WITH BLANK LINE
   100 PCT REMOVAL OVER 50 ACRES GIVES INITIAL IMPACT OF M6

    SHOULD THIS ENTRY BE FOLLOWED?
   YES
    LONG TERM EFFECT
    EVALUATE CHANGE IN BENTHIC COMMUNITY                    LEVEL  1 DUE TO
     CONSUMER REMOVAL                          = NOT EVAL.
    ENTER EFFECT ESTIMATE
(b)LO/M5/D/PR
    ENTER COMMENTS, TERMINATE WITH BLANK LINE
   CALCULATED ON THE BASIS OF AT LEAST 80 PERCENT RECOVERY OVER 50 ACRES
    -NOTE. COMMENTS CAN CONTINUE ON AS MANY LINES AS NEEDED.

    SHOULD THIS ENTRY BE FOLLOWED?
   NO
    IS INTERMEDIATE CONSUMERS
    SIGNIFICANTLY AFFECTED BY PEA CONSUMER REMOVAL
(c)$STOP
    CASI FILE WRITTEN, ENTRIES=    6 COMMENTS=     5
    TITLE= EXAMPLE CASE FOR TECHNICAL PAPER ON COMPUTER ASSISTED ESC
    LEVEL=    0 STATUS=    1
(d) DO YOU WANT TO CONTINUE WITH ANALYSIS ? YES/NO
```

Figure 3

RESUMING AN "OLD" ANALYSIS

```
GO:
 READ
 DATA BASE
MEDIUM SALINITY BAY SYSEM DIAGRAM MAY 1978, W. BROGDEN                JULY 6.3
 WITH   39 ATT, PEA=  13
NEW OR OLD ANALYSIS ?
OLD
 CASE FILE READ, ENTRIES=  172 COMMENTS= 109LEVEL=   3
 TITLE OF CASE
PILOT STUDY
 PREVIOUS CHANGES AFFECTING DISSOLVED OXYGEN  (a)          (b)     (c)
SEDIMENT DISSOLVED OXYGEN                  SH M6 D DF CAUSES SH M4 D PO 74
CURRENT ENERGY                             LO M5 I PR CAUSES LO M1   PR101
 AT PRESENT LEVEL DISSOLVED OXYGEN                           IS CHANGED BY
INTERMEDIATE CONSUMERS                     SH M6 D PR CAUSES SH M4 I PO110
TOP CONSUMERS                              SH M6 D PR CAUSES NOT EVAL. 118
INTERMEDIATE CONSUMERS                     LO M6 D PR CAUSES NOT EVAL. 125
TOP CONSUMERS                              LO M6 D PR CAUSES NOT EVAL. 133
HERBIVORES AND DETRITIVORES                SH M6 D PR CAUSES NOT EVAL. 147
TURBULENT ENERGY                           LO M7 D PR CAUSES NOT EVAL. 161
 EVALUATE CHANGE IN DISSOLVED OXYGEN                          LEVEL  3 DUE TO
  TOP CONSUMERS                      = SH M6 D PR
 ENTER EFFECT ESTIMATE
(d)SH/M1/ /PR
 ENTER COMMENTS, TERMINATE WITH BLANK LINE
TOP CONSUMERS ARE ONLY A MINOR OXYGEN CONSUMER

 SHOULD THIS ENTRY BE FOLLOWED?
NO
 EVALUATE CHANGE IN DISSOLVED OXYGEN                          LEVEL  3 DUE TO
  INTERMEDIATE CONSUMERS             = LO M6 D PR
 ENTER EFFECT ESTIMATE
```

5. Figure 2 shows the dialog resulting from a "NEW" response, and the analyst's entries are underlined. The first step required is giving a descriptive title which will be stored in the case file. The program echos the title and gives the analyst a chance to change it. Next, any number of comments may be entered (note that typing errors can be "backed" over with the backspace character, shift O on many keyboards).

   After the title and comments are entered, program goes through all of the possible PEAs for the ecosystem, and the operator must respond YES or NO to select those which occur in this analysis. For each PEA selected, a level zero entry in the data base is automatically created. Figure 2 shows the first few PEAs of the example analysis.

6. After the PEAs have been selected, the program proceeds as shown in Figure 2. The program will go through all possible level one attributes and the analyst must select those which are altered by the activity. For those which are selected, the analyst is required to evaluate both the short-term and long-term alterations. As shown in line (a) of Figure 2, the computer first gives the cause; in this case, the PEA consumer removal. The words NOT EVAL. appear in line (a) because PEAs do not have magnitude, etc. codes assigned.

   The analyst enters estimation codes as shown at (b) in Figure 2. If the program cannot recognize the codes, it will print out a review of the desired format and legal codes. Thus if the analyst has forgotten the format, he or she can just enter HELP, and the correct form will be displayed.

7. At any time after the initial PEAs (level zero) have been selected, the analyst can stop the program without loss of any information as shown in Figure 2 (c). After the word $STOP is entered, the case file is rewound and all of the entries, comments, etc. are written out. This operation affects only the local copy of the case file.

   It is wise to do this fairly frequently, so that if an unknown bug in the program causes it to end prematurely, the local file will contain the data needed to restart the analysis with minimum loss of effort.

   If the analyst answers NO to the question in line (d), the program terminates. The case file can now be saved on permanent file (see TAURUS documentation).

8. Figures 2 and 3 have shown the format for PEA and first-level attribute related operations. Figure 3 shows the resumption of an OLD analysis and the format used for attribute alterations at higher levels.

At this point in the analysis, the analyst is evaluating changes in dissolved oxygen at level 3. The program first reviews all changes in dissolved oxygen at lower levels. The example shown is read in the following way: At an earlier level, a short (M6 magnitude) decrease (definite) change in SEDIMENT DISSOLVED OXYGEN (a) caused a change in DISSOLVED OXYGEN summarized at (b); this is entry number 74 in the case file (c).

Next, changes or potential changes at level 3 are reviewed. In the example of Figure 3, six potential changes are identified, only one of which has been evaluated at this point in the analysis.

As shown at (d) in Figure 3, a blank may be entered in the estimation code if one of the codes can not be decided on.

9. After entering the estimation codes and any comments, the analyst is required to decide whether or not the entry should be "followed." The analyst must answer YES if the further consequences of an attribute alteration are to be worked out.

10. At the second and subsequent levels of analysis, the program collects all possible attribute changes, sorts them, and requires the analyst to evaluate the changes in the order of the attribute numbers. Thus, in the example of Figure 3, all of the potential changes in DISSOLVED OXYGEN at level 3 have been located, and the analyst will be required to evaluate all of them before moving on to another attribute.

    If only one "short" and/or only one "long" alteration is possible for a particular attribute, the program then moves on to the next. However, if more than one short or long change is possible (as in Figure 3, where three short and three long-term alterations are possible) the program gives the analyst the opportunity to summarize all changes with a given time span in a "collected" entry.

11. Whenever the analyst decides to create a collected entry, the program requires that it be evaluated just like the direct attribute alterations. The magnitude, direction, and probability should reflect the analyst's evaluation of the net change in the attribute for the given time span.

12. When all attribute changes at a given level have been evaluated, the program gives the analyst the following choices:

    0 - CONTINUE ANALYSIS
    1 - OUTPUT SUMMARY OF LAST LEVEL
    2 - $STOP - SAVE CURRENT CASE FILE
    3 - TRACE AND MODIFY

    The consequences of these choices are briefly described in the following sections.

13. CONTINUE ANALYSIS - If this option is chosen, the program looks at all of the attribute changes at the level just completed, and for those which were selected to be followed generates the next level of possible attribute alterations in the form of unevaluated entries. The number of new entries generated is displayed.

    If no new entries were generated, the analysis is assumed to be finished and the finished case file is written out. Otherwise, the program starts requesting evaluation of attribute alterations at the new level.

14. OUTPUT SUMMARY OF LAST LEVEL - If this option is chosen, the program gives the analyst two choices of output form and the chance to return to the display shown in (12) above. At this point, the analyst can review any or all of the entries in the level just completed. If the analyst elects to look at just the changes in an individual attribute, he must supply the program with the attribute number. This is the number assigned each attribute during the creation of the ESD data base. The output produced by the ALL ATTRIBUTES choice may be quite long and time consuming. For this type of review, the analyst should consider having the case file printed using the WRKSHT program.

15. The $STOP option causes the case file to be written out as discussed in (7) above.

16. The TRACE AND MODIFY option offers the only way in which an entry can be removed from the case file or modified. In order to use this facility, the analyst must know the number of the entry. This number can be obtained from displays such as that shown in Figure 3 (see note c) or from the output of the WRKSHT program. The program asks for this number and then prints the corresponding entry with its cause so the operator can verify that it is correct.

    If the analyst replies that this entry is to be modified, the program asks for the new estimation codes and comments. If the entry modified was at the level just completed, the program returns to the choices shown in (12). However, if the entry was at an earlier level, the program "backs up" to the earlier level and removes all entries dependent on the modified one. A message giving the new number of entries and level is given, and the computer returns to the choices shown in (12). NOTE: This procedure may renumber all entries following the one modified; therefore, if more than one entry is to be modified, start with the highest numbered entry first.

    CAUTION: Before using this option to modify entries at levels earlier than the level just completed, the analyst should use the $STOP option to create an updated local file copy of the case file

and save this in a permanent file. This routine has not been tested with all possible case file configurations and may produce unexpected modifications. Correct operation can be verified by running the WRKSHT program with the case file before and after performing the modification.

RUNNING THE WRKSHT PROGRAM

The purpose of this program is to print the contents of a case file in a format similar to the worksheet in the manual method. It is typically run from TAURUS after adding more material to a case file with the EVAL program. However, it requires no operator input.

1. Establish terminal connection with TAURUS if not already online.

2. If local working copies of the program, ESD data base, and case file do not exist, get them from a permanent file.

   Example:

   ```
   READPF(9290,BWORK,dbname,casename)
   ```

3. Execute the program as follows:

   ```
   BWORK(dbname,casename,OUTPUT)
   ```

   Naturally, the correct data base name and case file name must be substituted for dbname and casename.

   Only a few seconds of running time is required.

4. The output is now on the local file named OUTPUT. It can be printed at the computation center with the following commands:

   ```
   SPL(100) - for set page limit to 100 decimal
   RSF.     - for release system files
   ```

5. The case file and data base are not altered by this program.

3. DETAILED DOCUMENTATION OF THE SETUP PROGRAM

GENERAL DISCUSSION

The SETUP program is used to create and edit a data base representing in a simplified way the interactions depicted in an ecological system diagram. Operator input is through an interactive terminal. Output consists of a new (or modified) data base and, optionally, a summary of the data base formatted for a wide printer.

Program flow within the SETUP program is relatively simple. After a new data base title is entered, or an old data base is read in, the operator has an opportunity to enter new attribute names up to the limit of 50, which is adequate to describe all ESDs at the present time. Next, the operator can alter previous names to correct misspellings, etc.

In the initialization step, all elements of the ND matrix (see data base structure, Figure 4) were set to zero, signifying no interaction. Thus, the operator only has to set those elements in the matrix which represent a direct interaction. This is done by first selecting a row of the matrix, N. Now, working from the ESD, the ecosystem description, and his or her knowledge of the system, the operator selects each attribute which is directly altered by a change in attribute N. For each selected attribute, an integer value is entered, representing the type of interaction (see Figure 4). When through with a "row" the operator can enter zero to get back to the stage where another row may be selected. At this stage he can enter a zero to go on to PEA modification.

If PEA modifications are not desired, the final phase of the program is entered. In this phase, the options are:

0 = stop the program
1 = print the data base on the output file
2 = save the data base, replacing the previous version in the local file

If PEA modifications are desired, the operator is given a chance to add new PEA names, then to alter previous names. Finally, the operator can alter the ND matrix row by row, just as described above for the attribute-attribute interaction part of the ND matrix.

Figure 4

DATA STRUCTURE OF ESD DATA BASE, WITH PARTIAL EXAMPLES
FROM THE MEDIUM SALINITY BAY

row	NAME array	ND array
1	SUN	0000000000200000 -
.		
.		
.		
10	HEAT IN WATER COL	0000000002020000 . . . . .
.		
.		
.		
39	BIRDS	 0200001001000
.		
.		
.		
.		
.		
.		
.		
.		
.		
89	INORGANIC MATERIALS	. . . . .
90	CONSUMER REMOVAL	 001101011

DATA STRUCTURE OF ESD DATA BASE

The data base consists mainly of two arrays: the NAME array and the ND array. The ND array is dimensioned 90 rows by 50 columns. It is an integer array representing the interaction possibilities between attributes in an ecological system diagram plus the potential PEAs (primary ecological alterations). As presently dimensioned, up to 50 attributes can be handled, and the sum of attributes plus PEAs can be as high as 90. The ND matrix is interpreted as follows: if attribute "n" is altered, look at row n in the ND matrix. Columns with a zero represent attributes which are not affected directly by attribute n. Nonzero values indicate a possible interaction, thus if ND(n,k) $\neq$ 0 then a change in attribute n may directly cause a change in attribute k. If the value is 1, an increase in n causes a decrease in k. A value of 2 indicates that an increase in n causes an increase in k. A value of 3 indicates a relationship too complex to describe in this fashion.

At the present time, the EVAL program does not use the actual value recorded, only the fact that the value is nonzero. However, the presence of this value offers possibilities for future development, such as linkage with models or an expanded data base.

The PEA-to-attribute interaction possibilities are represented in a similar fashion, starting at the "bottom" of the ND array and working "up." Thus, the first PEA is in row 90, etc. A nonzero value indicates that an interaction is possible, but no significance is attached to the exact value.

The NAME array is dimensioned 90 rows by 4 columns, with the rows corresponding to those in the ND array. The four columns are used in A10 format to store 40 characters. The relationship of the NAME and ND arrays is shown schematically in Figures 5, 6, and 7.

Also included in the data base are two integer counters: NNAME = the number of attributes, and NPEA = the number of PEAs. The title of the data base is stored in 8A10 format in the NTITLE array.

SUBROUTINES USED WITH SETUP PROGRAM

SUBROUTINE RST(N)

In order to handle a conversational terminal in FORTRAN on the UT CDC6000, an end of file mark must be written after output to the terminal, and the file must be rewound. The file must also be rewound after input. This subroutine performs these tasks. It uses the convention that the input file is always TAPE5 and the output file is always TAPE6.

Figure 5

FLOWCHART FOR PROGRAM SETUP, PHASE I

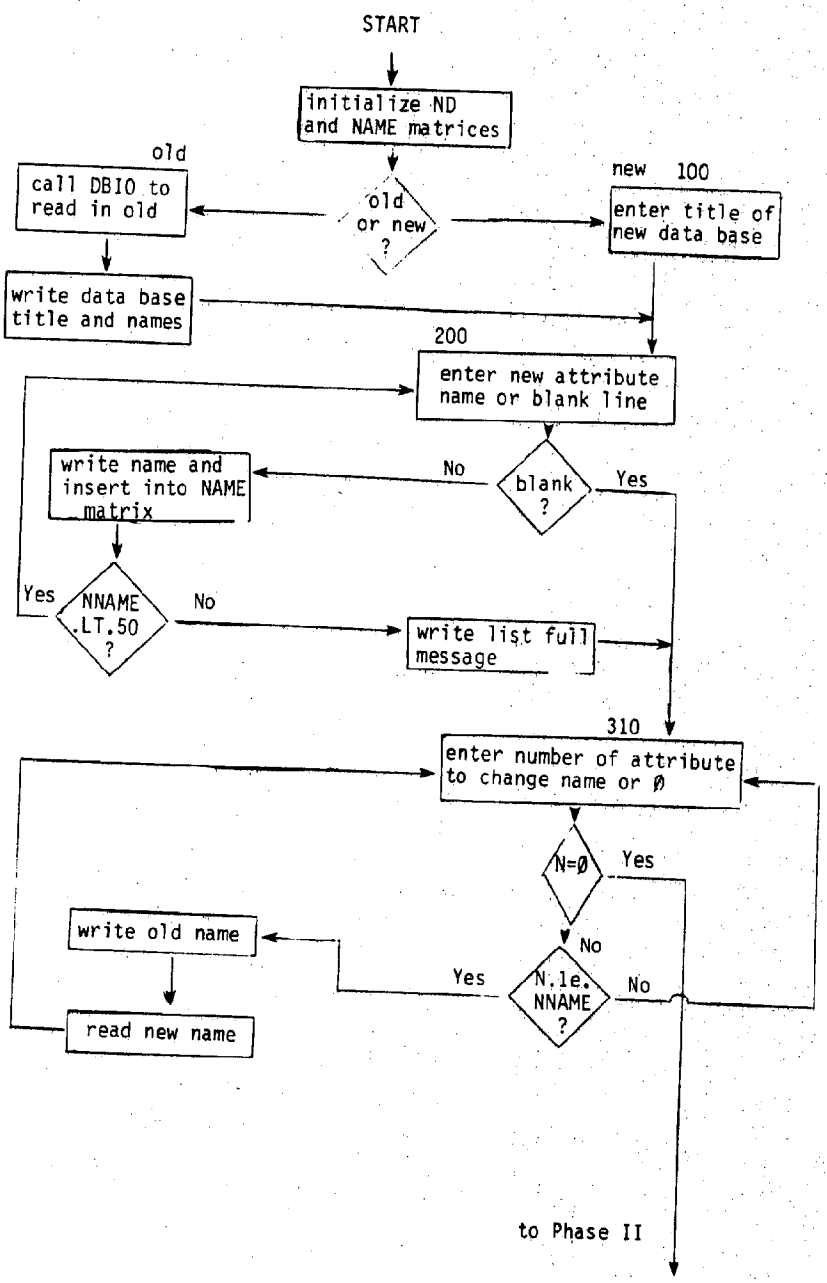


Figure 6

FLOWCHART FOR PROGRAM SETUP, PHASE II

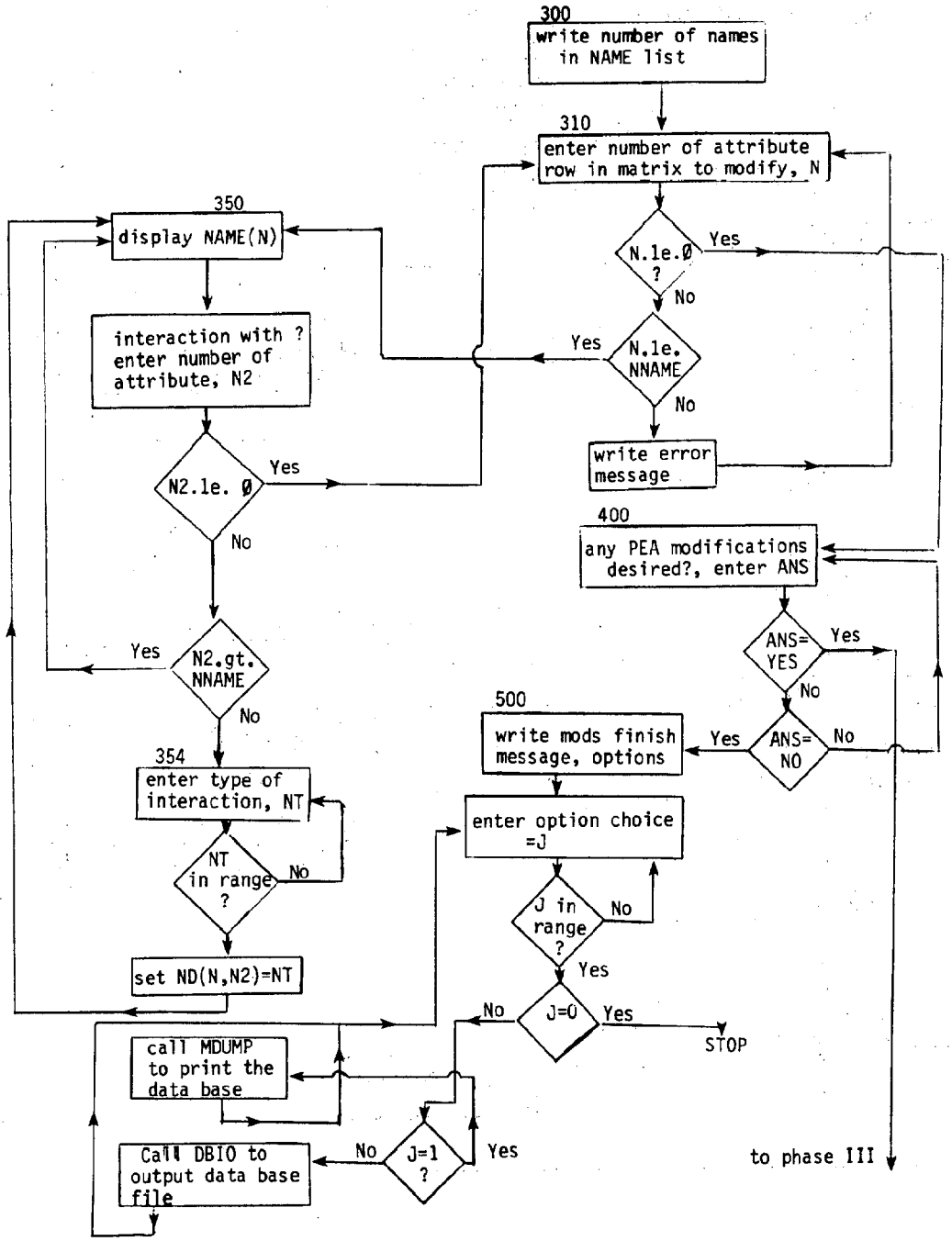
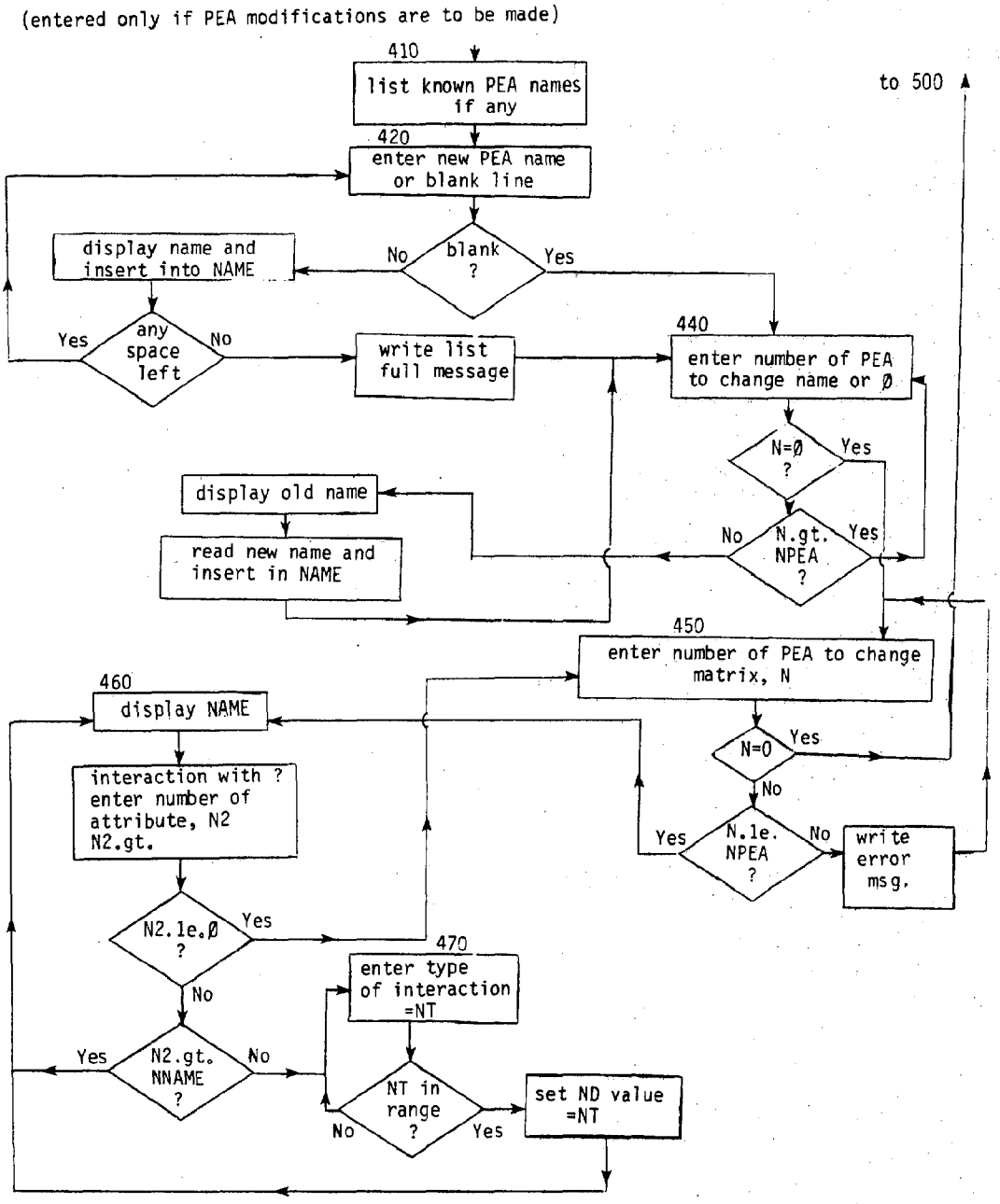


Figure 7

FLOWCHART FOR PROGRAM SETUP, PHASE III



SUBROUTINE INUM(I)

A simple formatted read cannot be used to get integer numbers from the operator because of the lack of error-handling alternatives. This routine reads the first 10 characters of an input line and turns them into an integer. If it can not recognize a character, it gives an error message and asks for a new number. Only numerals 0 through 9 plus blank can be recognized.

SUBROUTINE DBIO(N)

This routine rewinds TAPE8, the data base file, then reads (N=1) or writes (N=0) a data base file. The same routine is used in all programs using an ESD data base file to ensure compatibility.

SUBROUTINE MDUMP(N)

MDUMP(N) is a simple routine to output the data base to the conversational terminal or system printer, depending on the value of N. The output format improves the readability of the ND matrix by arranging it in columns of five.

SUBROUTINE PNAME(N)

After checking for N out of range, PNAME(N) prints the name corresponding to N from the NAME array. If the N is out of range, a message is printed and a new N is accepted.

PROGRAM LISTING

```
      PROGRAM SETUP(TAPE5,TAPE6,TAPE7,TAPE8)
C TTY IN, TTYOUT,  PRINTER OUT, DATA BASE
C BY W.R.BROGDEN  RPC INC. FOR TEXAS GENERAL LAND OFFICE
C  MAY, 1978
      COMMON/ / NNAME,NPEA,NAME(90,4),ND(90,50),NTITLE(8)
      INTEGER ANS,TNAME(4)
C INITIALIZE DATA MATRIX
      DO 12 J=1,90
      DO 11 K=1,4
   11 NAME(J,K)=10H
      DO 12 K=1,50
   12 ND(J,K)=0
      NPEA=0
      NNAME=0
```

COASTAL ZONE
INFORMATION CENTER

```
C
   14 WRITE(6,15)
   15 FORMAT(* MATRIX MODIFICATION PROGRAM*,/* OLD OR NEW> O/N*)
      CALL RST(6)
   18 READ(5,20) ANS
   20 FORMAT(8A10)
      CALL RST(6)
      IF(ANS.EQ.1HN) GOTO 100
C
      IF(ANS.NE.1HO) GOTO 14
C  HERE WITH OLD DATA BANK TO BE MODIFIED. READ FROM TAPE8
      CALL DBIO(1)
      WRITE(6,31)(NTITLE(J),J=1,8)
   31 FORMAT(* TITLE OF OLD DATA BASE *,/,2X,8A10,/,
     1 * PRESENT NAMES KNOWN *)
C
      DO 40 J=1,NNAME
      WRITE(6,37) (NAME(J,K),K=1,4),J
   37 FORMAT(5X,4A10,I4)
   40 CONTINUE
      GOTO 200
C
C NEW DATA BANK IF HERE
  100 WRITE (6,101)
  101 FORMAT(* NEW DATA BASE - ENTER TITLE *)
      CALL RST(6)
      READ(5,20)(NTITLE(J),J=1,8)
      CALL RST(5)
C
C START ASKING FOR NAMES
  200 WRITE(6,201) NNAME
  201 FORMAT(I5,* NAMES PRESENT*,/,* ENTER NEW NAME OR BLANK LINE*)
      CALL RST(6)
      READ(5,20) (TNAME(K),K=1,4)
      CALL RST(5)
      IF(TNAME(1).EQ.10H              ) GOTO 250
      NNAME=NNAME+1
      WRITE(6,220) NNAME,(TNAME(K),K=1,4)
  220 FORMAT(* NAME *,I3,2X,4A10)
      DO 230 J=1,4
  230 NAME(NNAME,J)=TNAME(J)
      IF(NNAME.LT.50) GOTO200
      WRITE(6,235) NNAME
  235 FORMAT(* LIST FULL *,I4)
```

```
  250 WRITE(6,251)NNAME
  251 FORMAT(* TO REPLACE EXISTING NAME ENTER ATTRIBUTE NUMBER 1-*,
     1 I4,/,* OR 0 TO GO ON TO NEXT SECTION*)
      CALL RST(6)
      CALL INUM(NT)
      IF(NT.EQ.0) GOTO300
      IF(NT.GT.NNAME) GOTO250
C
  260 WRITE(6,261) NT
  261 FORMAT(* ATTRIBUTE *,I4)
      CALL PNAME(NT)
      CALL RST(6)
      READ(5,20)(TNAME(K),K=1,4)
      CALL RST(5)
      DO 270 J=1,4
  270 NAME(NT,J)=TNAME(J)
      GOTO250
  300 WRITE (6,301) NNAME
  301 FORMAT(* TOTAL NAMES= *,I4)
  310 WRITE(6,311)
  311 FORMAT(* TO MODIFY MATRIX ENTER THE NUMBER OF *,/,
     1 * THE FIRST ATTRIBUTE OR 0 TO QUIT *)
      CALL RST(6)
      CALL INUM(N)
C N=SELECTED FIRST ATTRIBUTE
      IF(N.LE.0) GOTO 400
      IF(N.LE.NNAME) GOTO 350
C HERE WITH ERROR
      WRITE(6,331) N,NNAME
  331 FORMAT(* ENTRY*,I4,* TOO LARGE, MAX= *,I4)
      GOTO 310
C HERE WITH GOOD NUMBER
  350 CALL PNAME(N)
      WRITE(6,351)
  351 FORMAT(* INTERACTION WITH>*)
      CALL RST(6)
      CALL INUM(N2)
      IF(N2.LE.0) GOTO 310
      IF(N2.GT.NNAME) GOTO 350
      CALL PNAME(N2)
C
  354 WRITE(6,355) N,N2
  355 FORMAT(2I4,* INTERACTION TYPE≥ 0-3*)
      CALL RST(6)
      CALL INUM(NT)
      IF((NT.LT.0).OR.(NT.GT.3)) GOTO 354
      ND(N,N2)=NT
      GOTO 350
```

```
C  NOW DO PEA NAMES, ETC.
  400 WRITE(6,401)
  401 FORMAT(* PEA MODIFICATIONS ? YES OR NO *)
      CALL RST(6)
      READ(5,20) ANS
      CALL RST(5)
      IF (ANS.EQ.3HYES) GOTO410
      IF (ANS.EQ.2HNO) GOTO500
      GOTO400
C*****

  410 WRITE(6,411) NPEA
  411 FORMAT(I4,* PEA NAMES KNOWN*)
      IF(NPEA .EQ.0)GOTO420
      DO 416 J=1,NPEA
      JP=91-J
  416 WRITE(6,417) J,(NAME(JP,K),K=1,4)
  417 FORMAT(*PEA *,I3,X,4A10)
C
  420 WRITE(6,421)
  421 FORMAT(* ENTER NEW PEA OR BLANK LINE*)
      CALL RST(6)
  424 READ(5,20)(TNAME(K),K=1,4)
      CALL RST(5)
      IF(TNAME(1).EQ.10H            ) GOTO440
C HERE WITH NEW PEA NAME
      NPEA=NPEA+1
      JP=91-NPEA
C REMEMBER THAT PEA NAMES GO FROM THE TOP OF THE NAME AND NO MATRIX DOWN
      DO 430 K=1,4
  430 NAME(JP,K)=TNAME(K)
      WRITE(6,433) (NAME(JP,K),K=1,4),NPEA
  433 FORMAT(X,4A10,* IS THE *,I3,* PEA*)
      IF((NNAME+NPEA).LT.90) GOTO420
C HERE NAME MATRIX FULL
      WRITE(6,437)NNAME,NPEA
  437 FORMAT(* MATRIX FULL*,2I4)
  440 WRITE(6,441)
  441 FORMAT(* ENTER PEA NUMBER TO CHANGE SPELLING, ELSE 0*)
      CALL RST(6)
      CALL INUM(N)
      IF(N.EQ.0) GOTO450
      IF(N.GT.NPEA) GOTO440
      JP=91-N
      CALL PNAME(JP)
      CALL RST(6)
      READ(5,20)(TNAME(K),K=1,4)
      CALL RST(5)
      DO 444 K=1,4
  444 NAME(JP,K)=TNAME(K)
      GOTO440
```

```
C NOW TO MODIFY ND MATRIX
  450 WRITE(6,451)
  451 FORMAT(* TO MODIFY THE PEA-ATTRIBUTE INTERACTION MATRIX*)
  454 WRITE(6,455)
  455 FORMAT(* ENTER PEA NUMBER OR 0 TO QUIT*)
      CALL RST(6)
      CALL INUM(N)
      IF(N.LE.0) GOTO500
      IF(N.LE.NPEA)GOTO460
      WRITE(6,331) N,NPEA
      GOTO454
C HERE WITH GOOD NUMBER
  460 JP=91-N
      CALL PNAME(JP)
      WRITE(6,351)
      CALL RST(6)
      CALL INUM(N2)
      IF(N2.LE.0) GOTO454


      IF(N2.GT.NNAME) GOTO460
      CALL PNAME(N2)
  470 WRITE(6,471)
  471 FORMAT(* INTERACTION TYPE> 0-2*)
      CALL RST(6)
      CALL INUM(NT)
      IF((NT.LT.0).OR.(NT.GT.2)) GOTO470
      ND(JP,N2)=NT
C NOW GO BACK FOR MORE ATTRIBUTES THIS PEA
      GOTO460
  490 WRITE(6,491) NPEA,NNAME
  491 FORMAT(I4,* PEAS*,I4,* ATTRIBUTES*)
CCCCCCCCCCCCCCCCCCCCCCC
C  AFTER LAST MODIFICATION
CCCCCCCCCCCCCCCCCCCCCCC
  500 WRITE(6,501)
  501 FORMAT(* MODS NOW FINISHED, NOW CHOOSE*)
  504 WRITE(6,505)
  505 FORMAT(* 0=QUIT 1=PRINT MATRIX 2=SAVE DATA BASE*)
      CALL RST(6)
      CALL INUM(J)
      IF(J.LT.0)GOTO504
      IF(J.GT.2)GOTO504
      IF(J.EQ.0)STOP
      IF(J.EQ.1) GOTO600
C SAVE DATA BASE J=2
      CALL DBIO(0)
      GOTO504
C PRINT DATA BASE MATRIX ON OUTPUT FILE
  600 CALL MDUMP(7)
      GOTO 504
      END
```

```
C UTILITY SUBROUTINES FOR SETUP AND OTHER PROGRAMS
      SUBROUTINE RST(N)
C THIS ROUTINE IS NEEDED TO HANDLE INTERACTIVE I/O
      IF(N.EQ.6) END FILE 6
      REWIND N
      RETURN
      END
C


      SUBROUTINE INUM(I)
C TO GET A SINGLE INTEGER FROM INPUT ^ REWIND INPUT
      INTEGER A(10),NC(10)
      DATA (NC(K),K=1,10)/1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,
     1 1H8,1H9 /
    4 READ(5,7)(A(K),K=1,10)
    7 FORMAT(10A1)
      I=0
      DO 20 K=1,10
      IF(A(K).EQ.1H )GOTO20
C SKIP BLANKS
      DO 12 N=1,10
      IF(A(K).EQ.NC(N))GOTO14
   12 CONTINUE
C ERROR IF HERE, NON-NUMERIC
      WRITE(6,13) (A(J),J=1,10)
   13 FORMAT(* NON-NUMERIC IN *,10A1,* RE-ENTER*)
      CALL RST(6)
      GOTO4
   14 I=(I*10)+N-1
   20 CONTINUE
      RETURN
      END
C
C
```

```
      SUBROUTINE DBIO(N)
C N=1 READ,  N=0 WRITE DATA BANK OUT TO TAPE8
      COMMON/ /NNAME,NPEA,NAME(90,4),ND(90,50),NTITLE(8)
      REWIND 8
      IF(N.EQ.1) GOTO 100
C WRITE
      WRITE(6,1)
    1 FORMAT(* ENTER MOD NUMBER, 10 CHAR MAX*)
      CALL RST(6)
      READ(5,5) NTITLE(8)
      CALL RST(5)
    4 WRITE(8,5)(NTITLE(J),J=1,8),NNAME
    5 FORMAT(8A10,I4)
      DO 10 J=1,80
   10 WRITE(8,11) (NAME(J,K),K=1,4),(ND(J,K),K=1,NNAME)
   11 FORMAT(4A10,5X,50I1)
      WRITE(6,13)
   13 FORMAT(* WRITE*)
      WRITE(6,15)(NTITLE(K),K=1,8),NNAME
   15 FORMAT(* DATA BASE *,/,8A10,/,* WITH *,I4,* NAMES*)
      WRITE(6,21)
   21 FORMAT(* REMEMBER TO SAVE NEW DATA BASE*)
      CALL RST(6)
      RETURN
C
C  HERE TO READ
C
  100 READ(8,5) (NTITLE(J),J=1,8),NNAME
      IF(NNAME.GT.0) GOTO108
      WRITE(6,103)
  103 FORMAT(* INVALID, NO DATA BASE*)
      STOP1
  108 DO 110 J=1,80
  110 READ(8,11)(NAME(J,K),K=1,4),(ND(J,K),K=1,NNAME)
      WRITE(6,113)
  113 FORMAT(* READ*)
      WRITE(6,15)(NTITLE(K),K=1,8),NNAME
      CALL RST(6)
      RETURN
      END
C
```

```
      SUBROUTINE MDUMP(N)
C OUTPUT MATRIX TO DISPLAY OR PRINTER
      COMMON/ /NNAME,NPEA,NAME(90,4),ND(90,50),NTITLE(8)
C N=6 DISPLAY N=7 PRINTER
      WRITE(6,5)
    5 FORMAT(* MATRIX OUTPUT*)
      CALL RST(6)
      WRITE(N,11) NTITLE
   11 FORMAT(1H1,* DATA BASE TITLE *,/,5X,8A10)
      WRITE(N,13) NNAME
   13 FORMAT(* CONTAINS *,I4,* ATTRIBUTES*)
      WRITE(N,15)
   15 FORMAT(//)
      DO 30 J=1,NNAME
   30 WRITE(N,31)(NAME(J,K),K=1,4),J,(ND(J,K),K=1,NNAME)
   31 FORMAT(2X,4A10,I4,10(2X,5I1))
      CALL RST(6)
      RETURN
      END
C
C


      SUBROUTINE PNAME(N)
C PRINTOUT NAME NUMBER N
      COMMON/ /NNAME,NPEA,NAME(90,4),ND(90,50),NTITLE(8)
C
    5 IF(N.LE.0) GOTO100
      IF(N.GT.NNAME) GOTO100
      WRITE(6,7)(NAME(N,K),K=1,4)
    7 FORMAT(4A10)
      RETURN
  100 WRITE(6,101) N,NNAM
  101 FORMAT(2X,14,* OUT OF RANGE 1-*,I2,* TRY AGAIN*)
      CALL RST(6)
      CALL INUM(N)
      GOTO5
      END
```

4. DETAILED DOCUMENTATION OF THE LISTS PROGRAM

GENERAL DESCRIPTION

The purpose of the LISTS program is to print out the relationships contained in an ESD data base in a form convenient for error checking. LISTS is used only in conjunction with SETUP during creation or modification of an ESD data base. As shown in the flowchart (Figure 8), after reading in a data base, the program cycles through all of the attributes present. For each attribute, the subroutine INTER is called to form lists of "preceding" and "following" attributes. For attribute NA, preceding attributes are those which when changed directly cause a change in NA, while following attributes are those which are changed by a change in NA.

The majority of the program is occupied with correctly formatting the output of these lists. The variable LEFT contains the space left on the page at any time. The NPR( ) array contains the list of following attributes, with JFL the count. Since names are stored in 4A10 format, the LINE array is 12 words long to contain preceding, attribute NA, and following names. Figure 8 shows how either names or blanks are inserted into the LINE array.

PEA interactions are printed in a simple loop after the attribute interactions are finished.

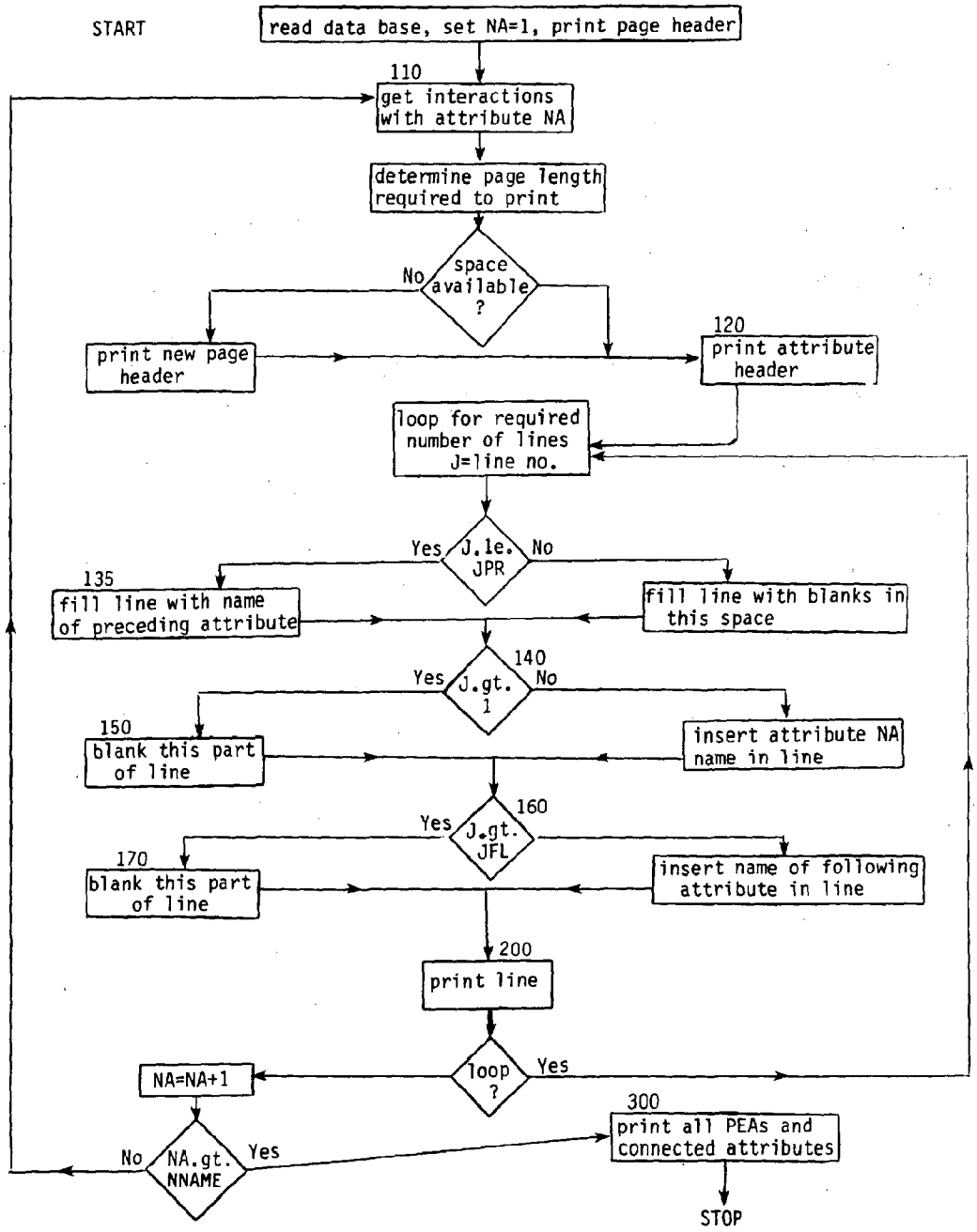
SUBROUTINES USED WITH LISTS PROGRAM

SUBROUTINE PRTITL

This routine simply prints the data base title as a page header.

Figure 8

FLOWCHART FOR PROGRAM LISTS



SUBROUTINE INTER(N)

This routine forms the two lists, NPR( ) and NFL( ), of "preceding" and "following" attributes. Nonzero values in row N denote following interactions, and nonzero values in column N cause alterations (precede) in attribute N.

SUBROUTINE DBIO(N)

This routine reads in the data base.

PROGRAM LISTING

```
      PROGRAM LIST5(TAPE8,OUTPUT,TAPE6=OUTPUT)
C   TO LIST ALL INTERACTIONS WITHIN AN ESD REPRESENTED BY
C   DATA BASE ON TAPE8 OR EQUIVILENT
      INTEGER BLANK,LINE(12)
      COMMON/INTR/NPR(80),NFL(80),JPR,JFL
C   NPR()=LIST OF PRECEDING ATTRIBUTES, TOTAL=JPR
C   NFL()=LIST OF FOLLOWING ATTRIBUTES, TOTAL=JFL
C
      COMMON/ /NNAME,NPEA,NAME(90,4),ND(90,50),NTITLE(8)
C
      DATA BLANK/10H          /
      WRITE(6,7)
    7 FORMAT(1H1,* INTERACTIONS LISTS*)
      CALL DBIO(1)
C TO READ DATA BASE IN FROM TAPE8
      DO 10 J=1,12
   10 LINE(J)=BLANK
      NA=1
C PRINT HEADER FOR PAGE
  100 CALL PRTITL
      LEFT=55
C GET INTERACTION
  110 CALL INTER(NA)
      JMAX=JPR
      IF(JFL.GT.JMAX) JMAX=JFL
      LNEED=JMAX+3
      IF(LNEED.LE.LEFT)GOTO120
C  LEFT IS THE NUMBER OF LINES LEFT ON PAGE, LNEED IS OBVIOUS
      CALL PRTITL
```

```
C NEW PAGE
      LEFT=55
C PRINT HEADER
  120 WRITE(6,121)NA
  121 FORMAT(/,*  PRECEEDING*,32X,*ATTRIBUTE *,I2,
     1 44X,*FOLLOWING*)
      LEFT=LEFT-3
      DO 200 J=1,JMAX
      IF(J.LE.JPR)GOTO135
C BLANK OR FILL WITH NAME
      DO 130 K=1,4
  130 LINE(K)=BLANK
      GOTO140
  135 DO 138 K=1,4
      J1=NPR(J)
  138 LINE(K)=NAME(J1,K)
C MIDDLE COLUMN
  140 IF(J.GT.1) GOTO150
      DO 145 K=1,4
  145 LINE(K+4)=NAME(NA,K)
      GOTO160
  150 DO 155 K=1,4
  155 LINE(K+4)=BLANK
C LAST COLUMN
  160 IF(J.GT.JFL) GOTO170
      K1=NFL(J)
      DO 165 K=1,4
  165 LINE(K+8)=NAME(K1,K)
      GOTO190


  170 DO 175 K=1,4
  175 LINE(K+8)=BLANK
C NOW OUTPUT LINE
  190 LEFT=LEFT-1
  200 WRITE(6,201)(LINE(K),K=1,12)
  201 FORMAT(2X,4A10,4X,4A10,4X,4A10)
C END OF LOOP, ANY MORE ATTRIBUTES ?
      NA=NA+1
      IF(NA.LE.NNAME) GOTO110
C NOW DISPLAY PEA LIST
  300 WRITE(6,301) NPEA
  301 FORMAT(*1  DATA BASE CONTAINS *,I3,* PEAS*)
      IF(NPEA.LE.0)STOP
      DO 400 J=1,NPEA
      IT=91-J
```

```
C REMEMBER PEAS GO FROM BOTTOM OF NAME AND ND - UP
      WRITE(6,303)J,(NAME(JT,K),K=1,4)
  303 FORMAT(*0 PEA *,I3,X,4A10,* INTERACTS WITH*)
      DO 360 N=1,NNAME
      IF(ND(JT,N).EQ.0)GOTO360
      WRITE(6,333)N,(NAME(N,K),K=1,4)
  333 FORMAT(5X,I5,2X,4A10)
  360 CONTINUE
      WRITE(6,361)
  361 FORMAT(///)
  400 CONTINUE
      STOP
      END
C**********
      SUBROUTINE PRTITL
C OUTPUT PAGE HEADER
      COMMON/ /NNAME,NPEA,NAME(90,4),ND(90,50),NTITLE(8)
      WRITE(6,7) (NTITLE(K),K=1,8)
    7 FORMAT(1H1,5X,8A10)
      RETURN
      END
      SUBROUTINE INTER(N)
      COMMON/INTR/ NPR(80),NFL(80),JPR,JFL
      COMMON/ / NNAME,NPEA,NAME(90,4),ND(90,50),NTITLE(8)
      JPR=0
      JFL=0
C GET PRECEEDING AND FOLLOWING ATTRIBUTES TO ATTRIBUTE N
      DO 100 J=1,NNAME
      IF(J.EQ.N) GOTO100
C SKIP SELF
      IF(ND(J,N).EQ.0)GOTO50
      JPR=JPR+1
      NPR(JPR)=J
   50 IF(ND(N,J).EQ.0) GOTO100
      JFL=JFL+1
      NFL(JFL)=J
  100 CONTINUE
      RETURN
      END
```

5. DETAILED DOCUMENTATION OF THE EVAL PROGRAM

GENERAL DISCUSSION

The EVAL main program operates in a straightforward fashion. There are two phases of operation, initialization and the cycle of evaluation/summary/interaction. The only complexity is introduced by the need to stop the analysis at any point and preserve the case file to that point, and conversely, to restart from old case files.

Initialization is controlled by the operator's response to the NEW OR OLD ANALYSIS question. In the case of a new analysis, statements 100 - 120 get the case title and comments, then call the PEA subroutine to create the initial (level 0) entries. In the case of an old analysis, the case file is read and the title is displayed, then the value of KSTATUS is used to control entry into the evaluation cycle.

For a new analysis or for cases stopped at this stage and restarted, the first-degree attribute changes are evaluated in EST1. This subroutine differs considerably from EST2 because it creates both short- and long-term effects entries and does not permit combination (collection) of entries. After the first level is past, this subroutine is not used.

For cases at or beyond the second level, the cycle is entered at the summarization stage or evaluation stage, depending on where the analysis was broken off. After summarization is complete at any level, INTRACT creates entries at the next highest level. If no new entries were created, analysis is stopped.

The evaluation program consists of a main program, EVAL, with a number of subroutines. Input to the program consists of the following:

TAPE8 - The ESD matrix for the ecosystem in which the activity to be evaluated occurs; this matrix is prepared by the program SETUP based on the ecological system diagram.

TAPE3 - The "case" file; this file is input only when an old case evaluation is being resumed. It is created only by the evaluation program at the operator's command. This file stores all of the data on the evaluation process, since the analysis process may take much more time than is typically available in one session with the computer.

TAPE5 - Operator input, assigned by the system to the TTY or conversational terminal file.

Output from the program can occur on three files:

TAPE6 - Assigned to output to the conversational terminal.

TAPE3 - The "case" file is output only if the operator selects the $STOP option. This can be done any number of times during the running of the program, with the most recent version always replacing the previous one.

OUTPUT - This is provided for certain types of system error messages and may be used in future developments to provide various printed summaries. Note that OUTPUT is a "system" file name and will be printed on the high-speed printer at the computation center if desired.

The general relationship of the main program to the various subroutines is shown in the following flowcharts (Figures 9, 10, and 11).

MAJOR VARIABLES WITHIN THE EVALUATION PROGRAM
AND SUBROUTINES - LISTED BY COMMON BLOCK

NOTE: All are type integer.

COMMON block ESD contains the ESD matrix information:

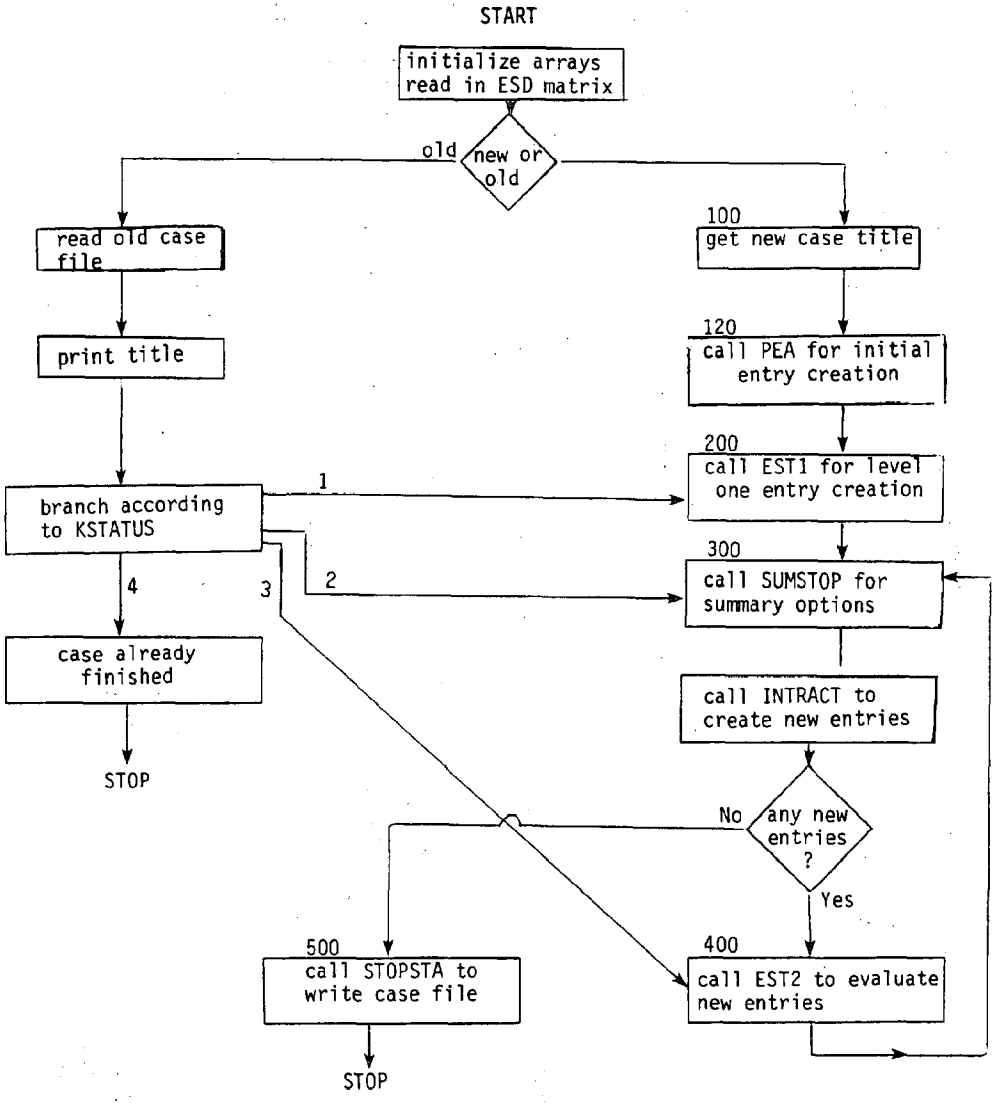
NNAME	number of attribute names in the matrix
NPEA	number of PEAs
NAME(90,4)	names of attributes and PEAs in 4A10 format
ND(90,50)	interaction matrix showing which attributes are connected (see description of SETUP program for more details)
NTITLE(8)	the title of the ESD matrix in 8A10 format

COMMON block EXEC contains major index counters:

NIEN	number of entries in IEN, the case file
NICOM	number of comments in ICOM
LEVEL	current working level: 0 for PEA, 1 for first-degree attributes, etc.
KSTATUS	used to control resumption of evaluation of an old case file
ICTRL(10)	used to contain loop counters during storage of a case file and to initialize counters during resumption of an old case

Figure 9

OVERALL FLOW WITHIN THE EVAL PROGRAM



Note that there are other stopping points within subroutines.

Figure 10

CALLING RELATIONSHIPS BETWEEN THE MAIN PROGRAM AND SUBROUTINES

The RST subroutine is called by most other subroutines which do input or output to the conversational terminal. SORT also calls the TSORT routine, a library routine in the UT CDC system.

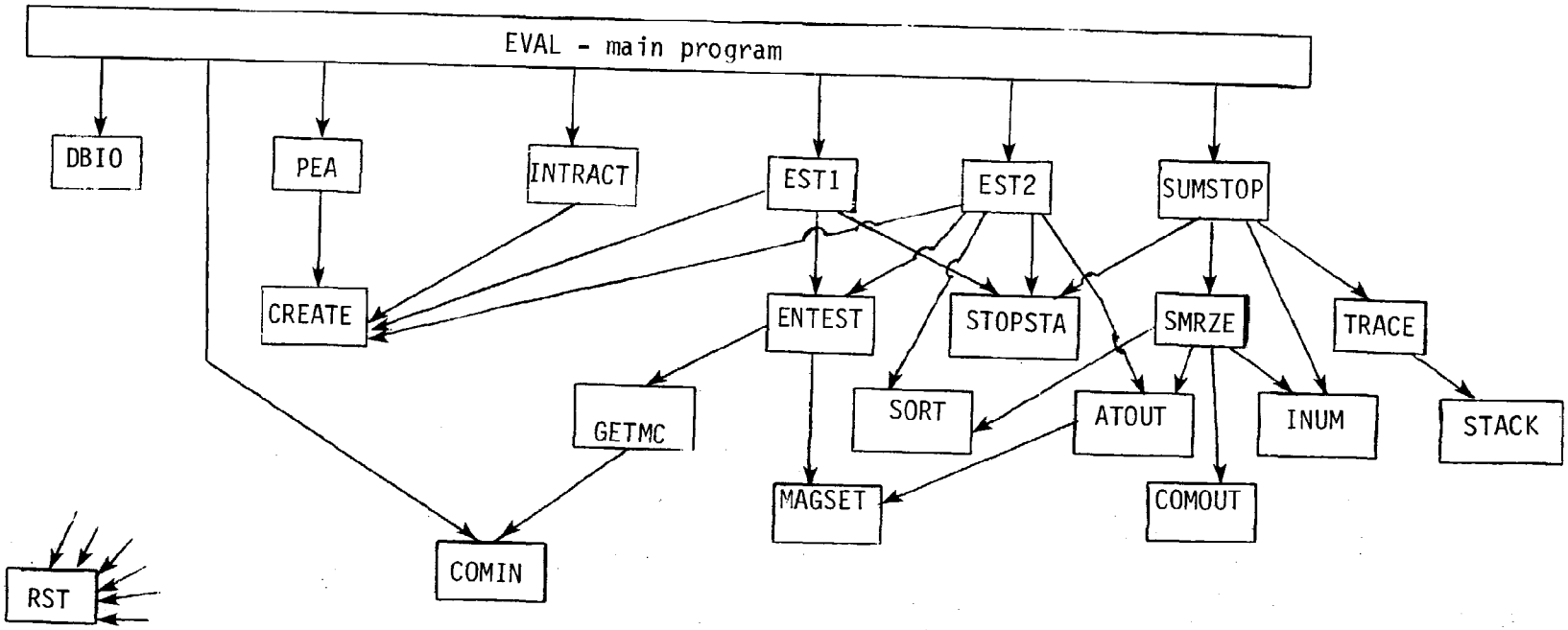
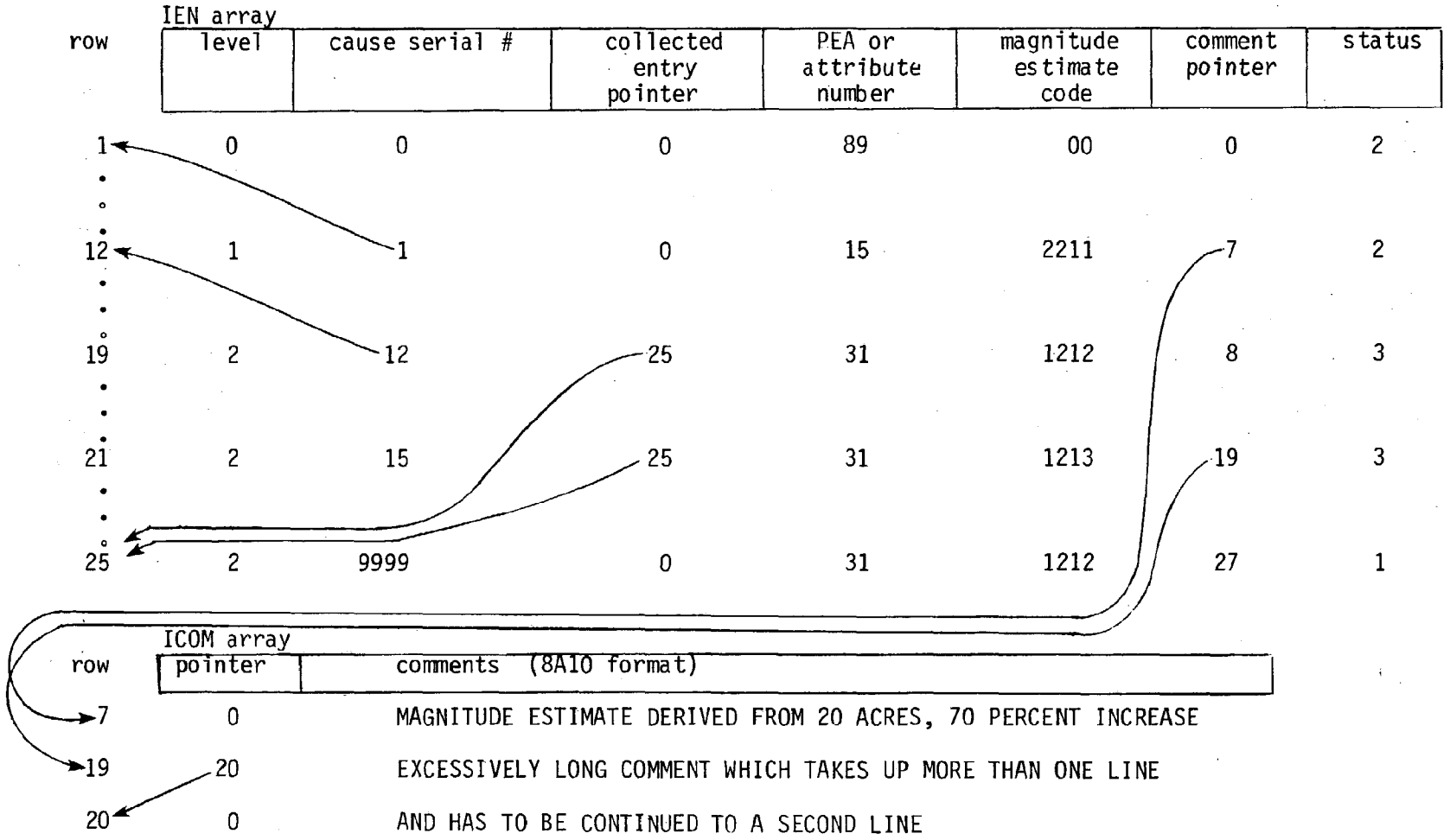


Figure 11

DATA STRUCTURE OF CASE FILE

The case file is composed of two arrays, IEN or entry array, and ICOM or comment array. Linkages within and between these arrays provide the structure and represent the decisions made in the analysis process. The ecosystem diagram data base is also linked to the entries through pointers to the names of attributes and PEAs.



blank COMMON — used for the major case file arrays; note that by providing a substantial area of blank common, less core space is required during loading of the program

IEN(400,7) — each entry in IEN represents a single decision about an attribute alteration as follows:

- (,1) — the level of decision (0 for PEA, etc.)
- (,2) — the serial number of the entry which leads to this one; except for PEAs it is 0 and for collected entries it is 9999
- (,3) — if a collected entry is created to represent the combined effect of several alterations to a given attribute at a given level, this number points to the collected entry; otherwise 0
- (,4) — number pointing to the attribute or PEA name in the NAME array
- (,5) — the importance estimate coded into a four-digit integer in the order time/magnitude/direction/probability
- (,6) — pointer to the comment array, ICOM( ), if a comment has been attached to this entry; else 0
- (,7) — status of entry, used to control generation of next higher level, etc. as follows:

0 = do not follow	1 = must follow
2 = already followed	3 = collected into another entry
9 = not yet evaluated	

ICOM(400,9) — used for comments attached to entries as follows:

- (,1) — if this comment is followed by additional comment lines, the next one is pointed to by this integer; if no follower, this is 0
- (,2-9) — the actual comment in 8A10 format, set equal to 0 at initialization

INDEX(400) — index is used to contain the result of sorts done on the IEN entry array; instead of moving the IEN array around, the pointers in INDEX are moved

VARIABLES IN COMMON BLOCKS WHICH
DO NOT APPEAR IN THE MAIN PROGRAM

COMMON block MAGWDS — used to share constants between subroutine MAGSET and GETMC (see subroutine documentation)

COMMON block STAK — used only by the TRACE and STACK routines

- ISP — current stack pointer
- ISTKMX — maximum legal value of stack pointer
- ISTK(200) — array used as stack

GENERAL DISCUSSION OF OTHER VARIABLES

In most cases, J and K are used as loop counters in DO loops. The integer variable ANS is used to read in replies from the conversational terminal. See individual subroutine discussions for more details.

MAJOR SUBROUTINE DESCRIPTIONS

SUBROUTINE PEA

This routine runs through the list of potential PEAs contained in the ESD matrix and asks the operator's opinion as to whether or not each PEA occurs to a significant extent. If the answer is YES, then a level 0 entry in the IEN table is CREATED with the name of the PEA and status=1 (for "must follow"). Since this procedure is very short and occurs right after the initiation of a new case, it cannot be interrupted by a $STOP command and later restarted like other major routines.

The only tricky point is that since the PEA names start at the high end of the NAME and ND arrays, the temporary variable JP has to be used to point to these arrays. After the last one has been evaluated, the actual number of entries created is reported.

SUBROUTINE STOPSTA

This subroutine stores the case file on tape 3 (which is equated to a named case file when the program is started).

Record	Contents
1	various counters in 1415 format
2 - NIEN+1	IEN array by rows, 715 format
NIEN+2 - NIEN+NICOM+1	ICOM array by rows, 17,3X,8A10 format
NIEN+NICOM+2 - NIEN+NICOM+1 + 1 rec. per 15 integers	INDEX array in 15I5 format

This subroutine also gives the operator an option of stopping the program after the case file is written or continuing the analysis.

SUBROUTINE INTRACT(NST,LEV)

This routine searches all entries in IEN for entries at level = LEV with status (IEN(,7)) set = 1 for "must follow." When one is found, the attribute altered, N, is used as an index into the ESD matrix ND to determine which attributes might be changed at level LEV+1. New entries are created for each attribute which might be changed. These entries have status = NST. In actual use, NST is always = 9 for "not yet evaluated."

SUBROUTINE EST1

Figure 12 is the flowchart for this subroutine. EST2 controls the generation of level 1 entries from the PEA entries (level 0). Loop control variables can be saved and restored if the analysis is stopped and the case file read in again later. This facility makes the routine somewhat more complex in the initialization of the loops. In the normal case, this routine is called just after the execution of the PEA subroutine, so the number of PEAs = NIEN (line 100) = LPEA. Major variables specific to this routine are:

LPEA - the number of PEA entries (level 0), used as a maximum
JIPEA - counts PEAs in the major program loop
JA - counts attributes in the secondary loop.
JP - points to the PEA name

In the major loop, the PEAs are gone through one at a time. For each PEA, the ND matrix (representing the ecosystem diagram) is examined to determine which attributes might be altered. The operator has to determine which ones should be followed as first-degree attributes (line 320, etc). The operator also has the option of entering $STOP, in which case the loop counters are transferred to the ICTRL array to be saved by the STOPSTA routine.

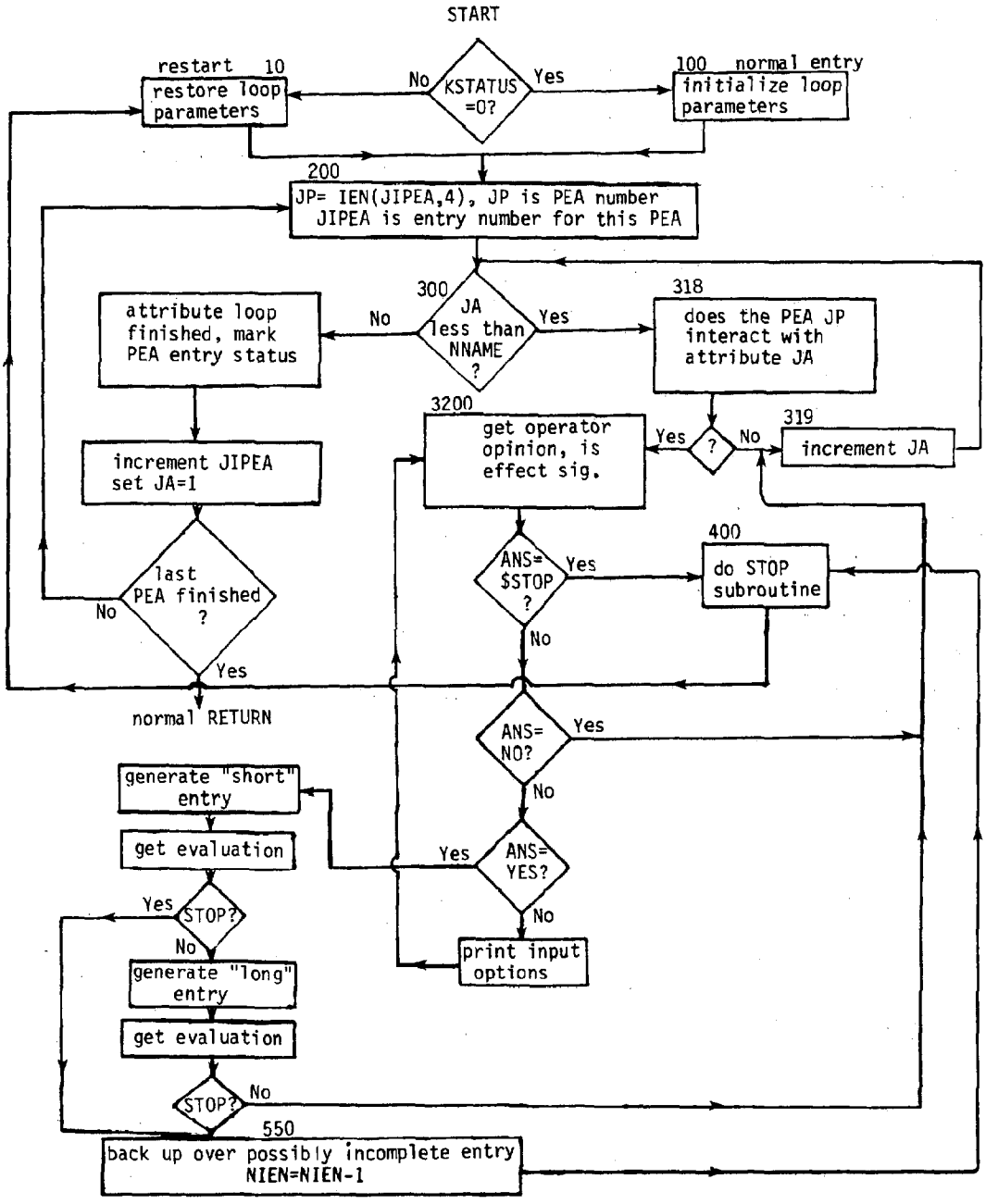
When the operator says that an attribute is significantly affected by the PEA, both a long-term and a short-term entry are generated (by CREATE) and the operator is asked for a significance estimate (by ENTEST).

SUBROUTINE SUMSTOP

This routine is called after a new level of analysis has been finished. It summarizes the number of entries at each level and gives the operator a choice of several options. Options are chosen by entering a number. Nonnumerical inputs are automatically rejected by subroutine INUM, while out-of-range numbers are rejected within this routine. Present options are:

Figure 12

FLOWCHART FOR SUBROUTINE EST1



0 = continue analysis, causes return to main program

1 = call SMRZE for a summary of level just finished

2 = set KSTATUS=2 and call STOPSTA to cause case file to be written

3 = trace and modify option is currently under development

Branches within the program are provided for options 4 and 5 but are not implemented.

SUBROUTINE EST2

This routine controls the input of importance estimates and comments on entries at level 2 and higher, with output of information on all preceding changes and changes at the present level before the operator is required to estimate the significance of a change in an attribute. This program consists of three phases. In the first phase (Figure 13), the attribute counter is initialized, the entries are sorted by attribute number, and the entries are searched for unevaluated entries. If an unevaluated entry is found for a given attribute, the second phase is entered.

The second phase (Figure 14) is entered only if at least one entry concerning attribute NA has not yet been evaluated. First, entries which caused attribute NA to be changed at lower levels are summarized. Next, entries at the present level, whether evaluated or not, are displayed. Next, the operator is asked to evaluate each change. If more than one "short" or "long" term change in attribute NA has occurred at this level, the third phase is entered.

In the third phase (Figure 15), the operator is given a chance to cause a "combined" entry for attribute NA to be created. If he chooses to create one, the operator evaluates the net change in attribute NA, and the entries leading to the combined entry are appropriately labeled.

SUBROUTINE TRACE

This subroutine is provided to permit the operator to return to a previous entry and modify it. Two clearly different situations may occur; modification of an entry on the present working level or modification of an entry on a lower (earlier) level. The first case is quite simple because only the entry itself need be changed. The second case is distinctly nontrivial because other entries may be affected.

Figure 13

FLOWCHART FOR SUBROUTINE EST2, PHASE I

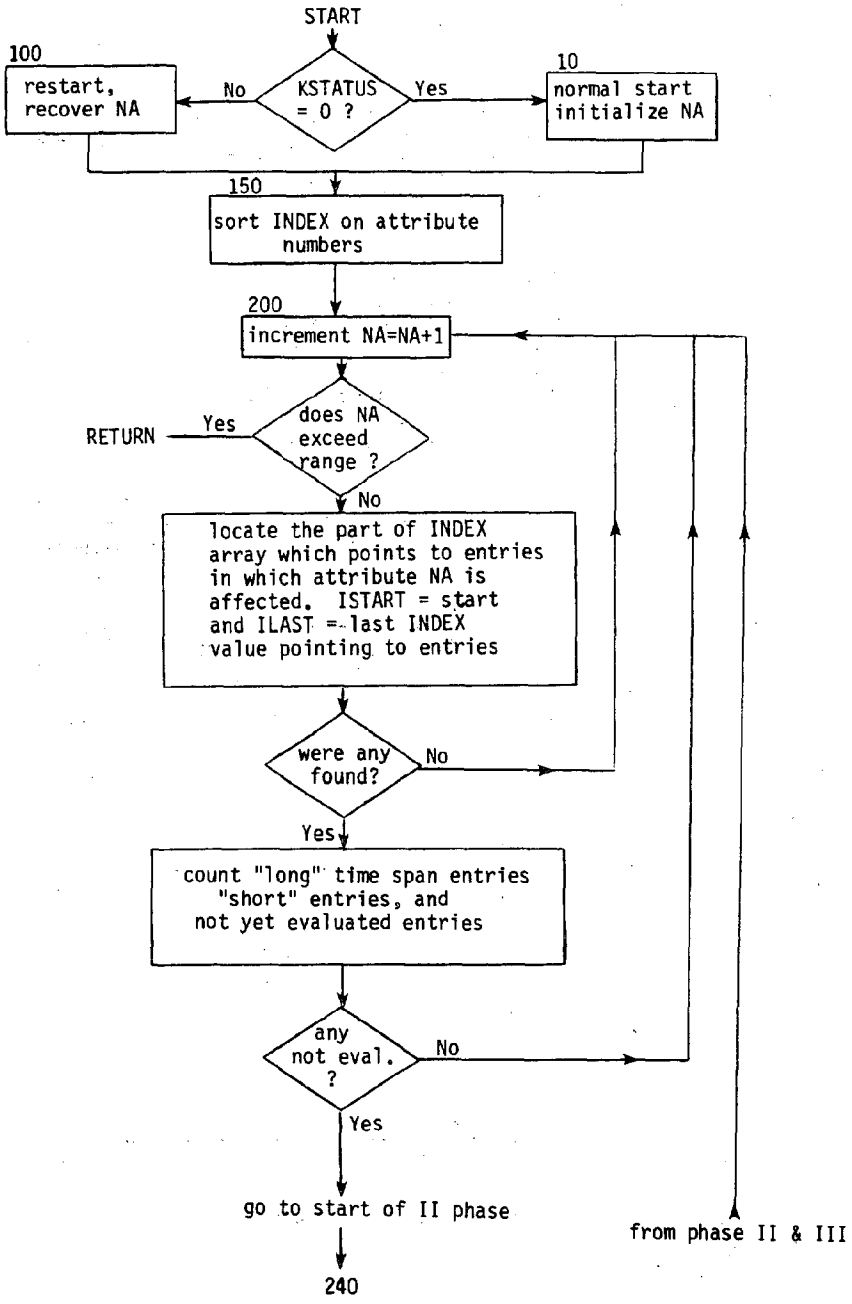


Figure 14

FLOWCHART FOR SUBROUTINE EST2, PHASE II

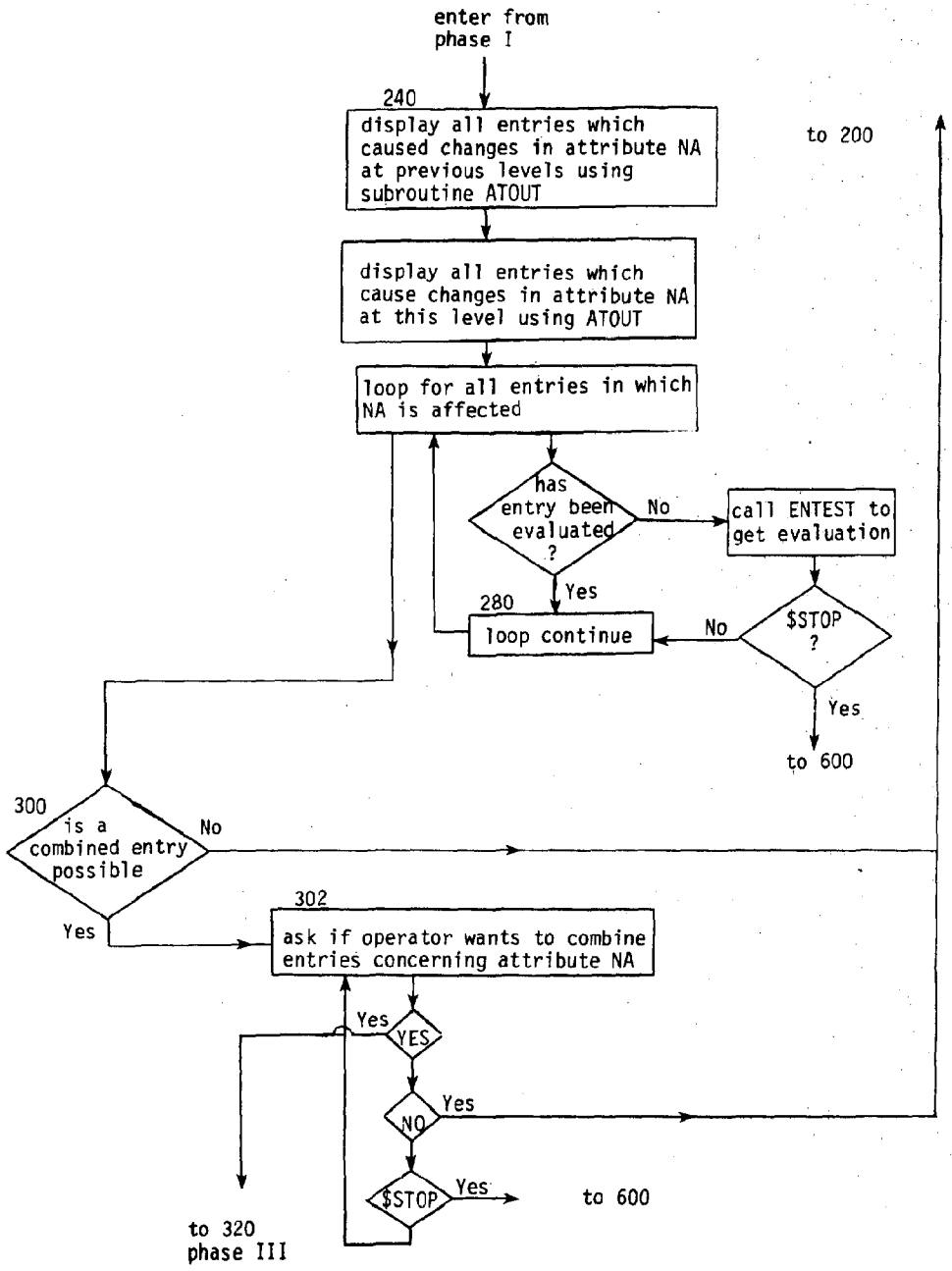
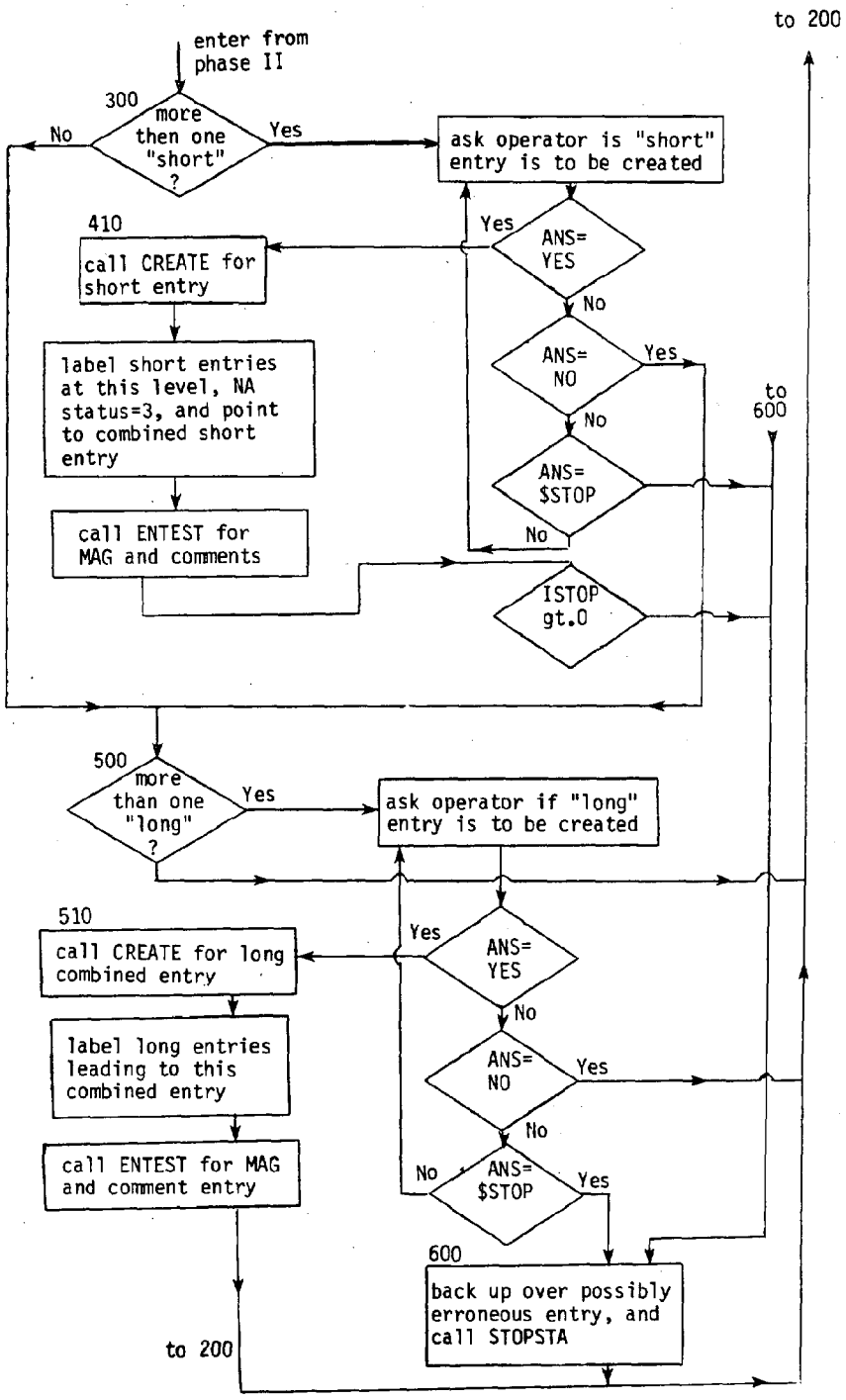


Figure 15

FLOWCHART FOR SUBROUTINE EST2, PHASE III



Referring back to Figure 11, it can be seen that entries can be linked by two pointers, the cause serial number and the collected entry pointer. The TRACE subroutine follows these pointers to trace and remove the effects of an attribute alteration from the case file so that the consequences of a modified entry can be determined properly.

The TRACE subroutine can be divided into four phases: (1) getting the number of the entry to be removed, (2) modifying the entry, (3) tracing and removing the consequences of the entry to permit reevaluation, and (4) recovering space in the IEN array.

Figure 16 shows the first phase, which consists simply of getting a number from the operator, verifying that it is in the proper range, and then displaying the entry to double check that this is the one the operator wished to change.

In phase II (Figure 17), the entry IEN(NCAUSE) is reentered using the ENTEST subroutine. However, the path taken depends on whether or not the entry NCAUSE is on the present LEVEL. If it is, there are no other consequences, and control returns to the calling program.

If the entry NCAUSE is at an earlier level, the third phase is entered (Figure 17). This phase uses a stack to accumulate pointers to the entries which are connected in some way to entry NCAUSE. The STACK subroutine which is called is the same one that is used in the WRKSHT program. Starting with a "seed" of those entries directly dependent on NCAUSE, the stacking process continues until the top entry has no further consequences. This entry is then reset to zero, the stack counter decremented, and the stacking loop reentered. Note that since entries are being zeroed out, calls to STACK will eventually result in no new additions even if the entry N originally had a number of dependent entries.

After the stack has been emptied, the space used by the zeroed-out entries is reclaimed by the final phase of the subroutine (Figure 18). This phase looks through the IEN array, and whenever a zeroed entry is encountered, it finds the next filled entry and moves it up. This requires readjustment of all pointers to the moved entry. Finally, the new number of entries, NIEN, is determined, and the LEVEL is adjusted so that when the main program is rejoined, the operator will have to trace out the altered consequences of entry NCAUSE.

SUBROUTINE ENTEST(JENT,ISTOP)

This routine is called with JENT pointing to an entry which needs to be evaluated. It displays the cause, asks for an evaluation, and comments through the GETMC routine. It finally asks the operator whether the entry is to be followed (i.e., if the entry is to generate more entries at the next level). Provision is made for entering $STOP to stop analysis and return to the calling routine with ISTOP=1.

Figure 16

FLOWCHART FOR SUBROUTINE TRACE, PHASE I

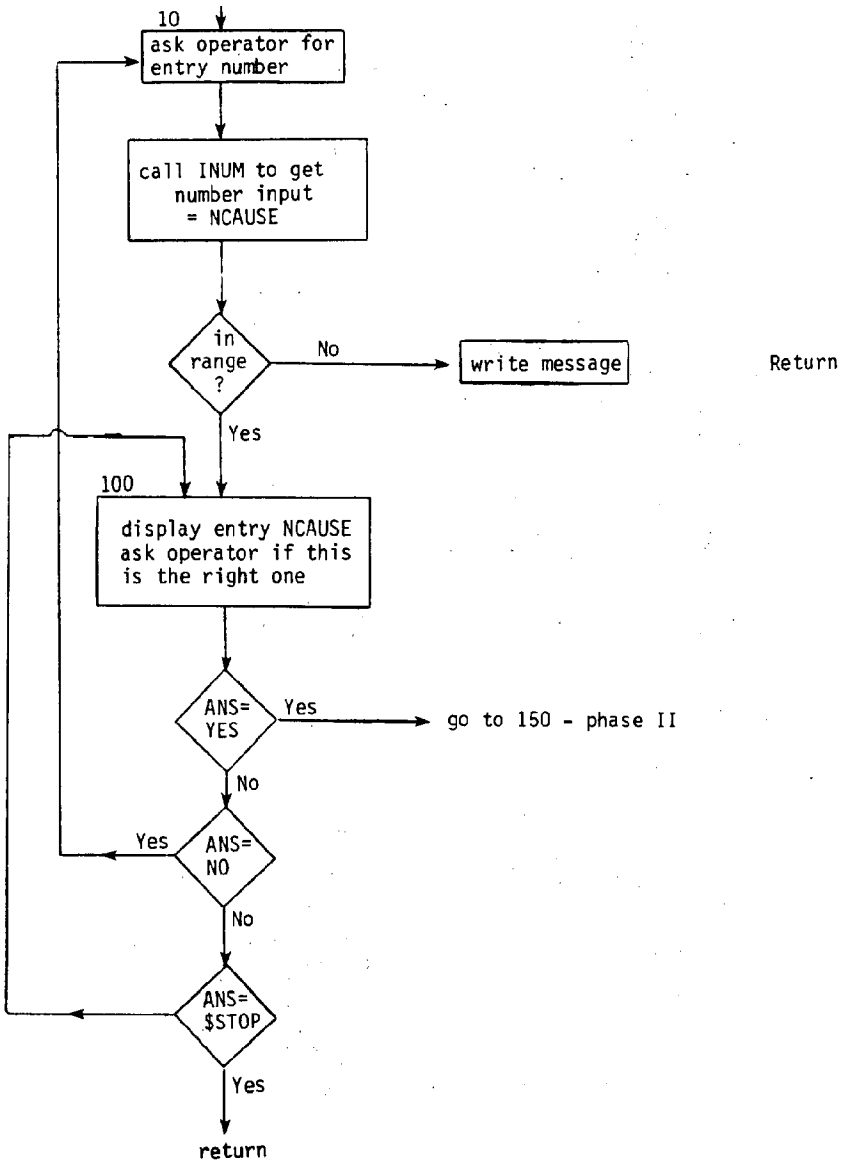


Figure 17

FLOWCHART FOR SUBROUTINE TRACE, PHASES II AND III

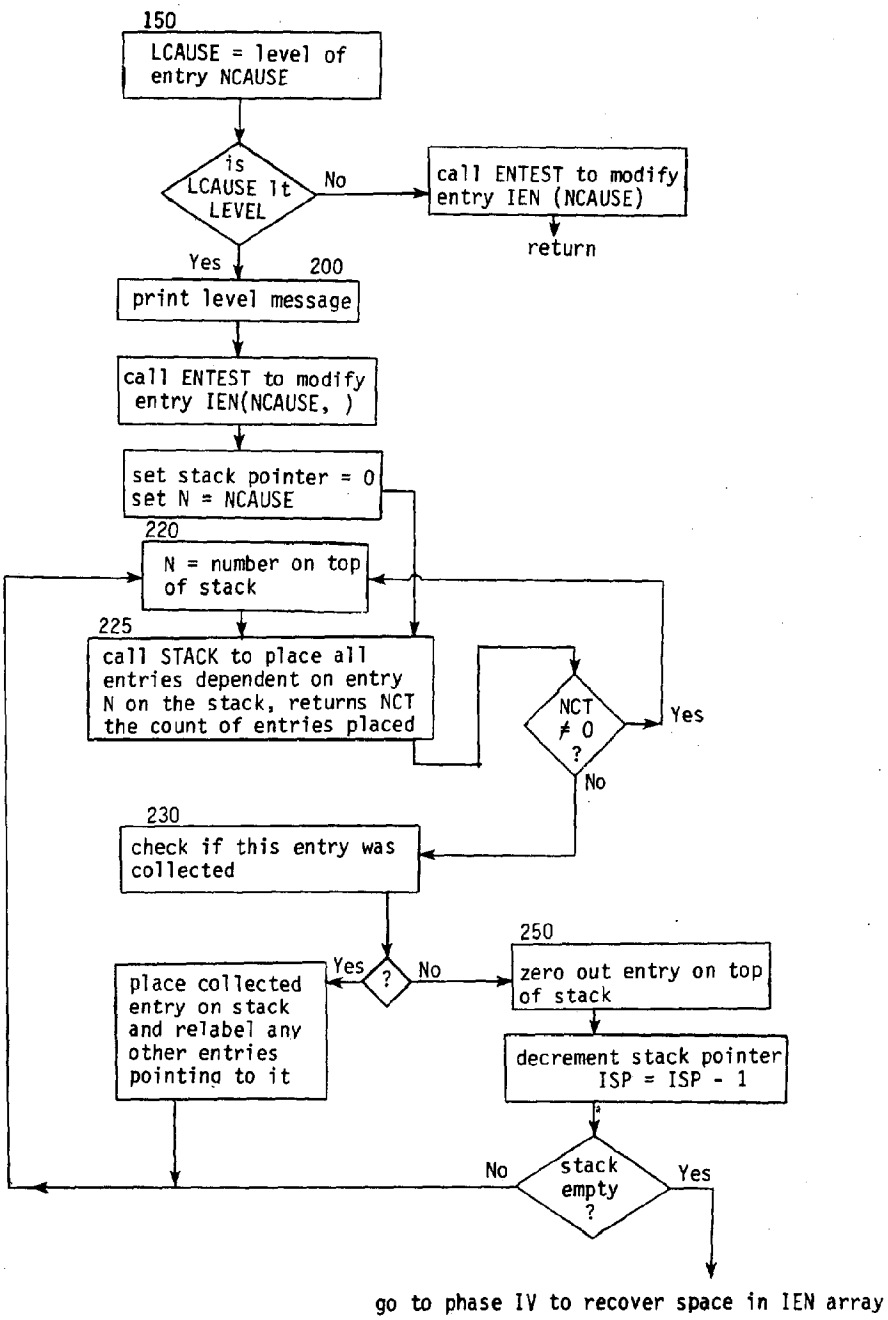
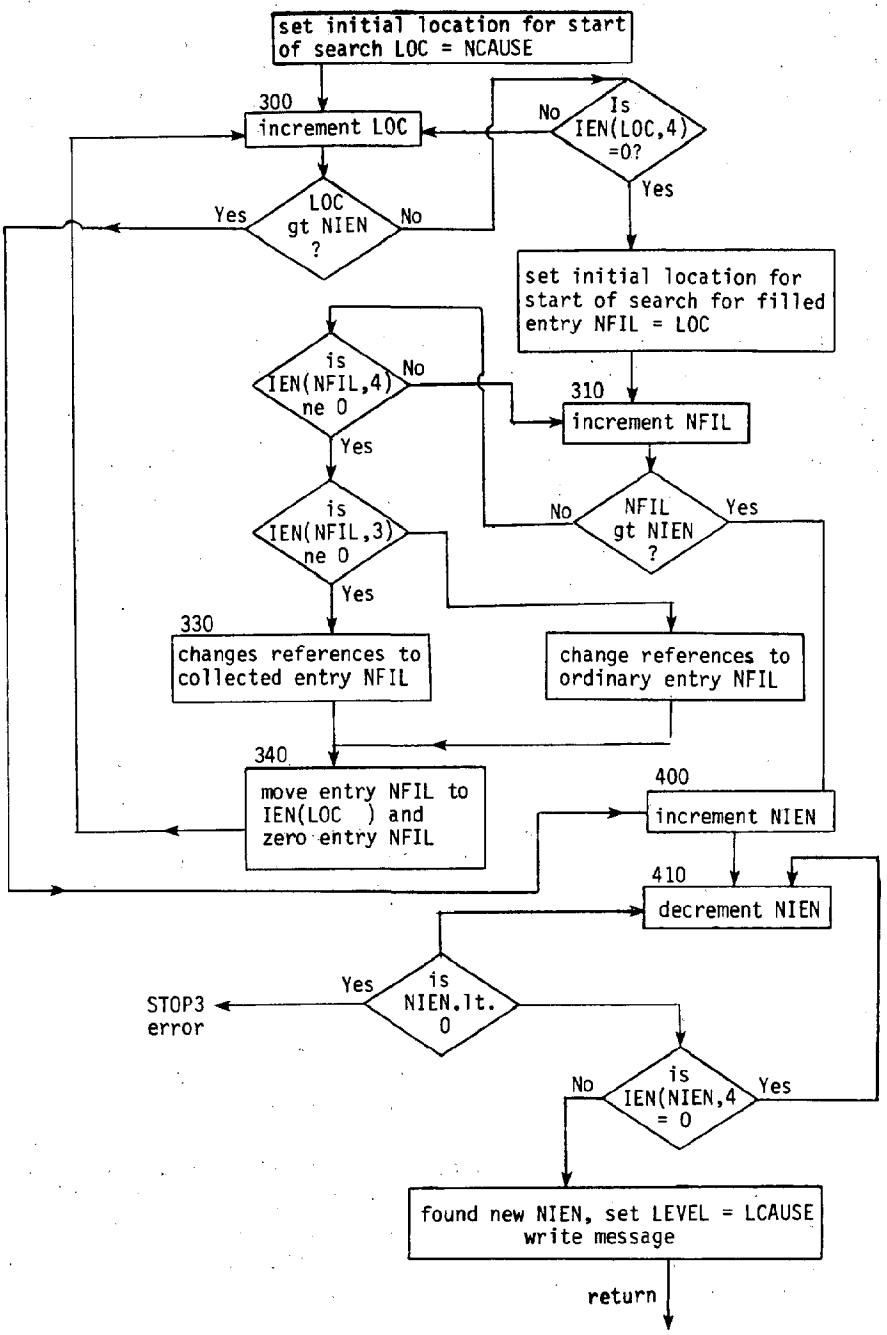


Figure 18

FLOWCHART FOR SUBROUTINE TRACE, PHASE IV



Entries which are to be followed are labeled 1 in column 7, and entries which are not to be followed are labeled 0.

SUBROUTINE GETMC(ISTOP,LINK,IMAG)

The routine is entered with LINK pointing to an entry, and it returns the time/magnitude/direction/probability estimate in IMAG as a four-digit code. It also passes LINK on to the COMIN routine for comment input. The complexity in this routine is necessitated by the need to recognize the code abbreviations in the input string. This is done by reading the operator reply in A1 format, locating the slashes and the end of the entry, then using ENCODE to insert the characters into an A3 format which is then compared with the legal abbreviations. If the input cannot be recognized, the program gives the correct syntax and abbreviations to the operator for another try.

SUBROUTINE SORT(JI,ITYPE)

The purpose of SORT is to create an array of pointers to the entries (IEN array) corresponding to alphabetical or numerical order of a selected element of the IEN array. For instance, this routine is called by EST2 to create a list of entries in order of attribute number. The INDEX array is used for the list of entries and the TEMP array is used as a sorting key. Sorting is actually done by the library routine TSORT. Provision is made for using a previously created order in INDEX to form the sorting array. This provision has not been tested and is not used in the present program, but it might be useful in sorting on a combination of elements in IEN.

SUBROUTINE ATOUT(JT)

This routine displays the cause of entry JT in terms of the attribute which was altered at a lower level as the result of a PEA or as a collected entry. The correct format is chosen first determining if entry JT is a collected entry (in which case IEN(JT,2)=9999), then, if it is not a collected entry, determining if the entry pointed to by IEN(JT,2) is a PEA.

SUBROUTINE COMIN(LINK,NCNT,ISTOP)

The only way comments get into the ICOM array is through this routine, except for the first comment which is the title of the case. It is called with LINK pointing to the entry in IEN which is to be linked with the comment. On return, NCNT equals the number of comments entered, and ISTOP equals zero unless the word $STOP has been entered. Any number of comments can be entered on one call to COMIN. The first are linked back to the preceding comment as described in Figure 11. Exit from this routine occurs only after a blank line or the word $STOP is entered.

SUBROUTINE COMOUT(NC,NF)

This routine displays a comment NC and all succeeding comments according to a format selected by NF. At the present time, only one format is provided. Note that NC is modified by the routine.

SUBROUTINE SMRZE

This routine can display previous entries at the present level in two different forms, with provision for adding more. The first action taken is to call the SORT routine to sort the entries based on magnitude code. The magnitude code is set up so that the resulting order will be: long before short, high magnitude before low, increase before decrease, and definite before probable.

The two options presently available are 1=all attributes and 2=selected attributes. Return from this routine is accomplished by selecting option 0. If option 1 is selected, N is cycled through all attributes by statements 80 and 90. The DO loop terminated by 94 is used to skip over values of N for which no entries at the present level exist. If option 2 is selected, the INUM routine is used to request the attribute number from the operator.

Statements from 110 to 140 output a header, then look through all of the entries and print the cause, the estimation codes, and the comments for entries at the present LEVEL and attribute N.

UTILITY SUBROUTINE DESCRIPTIONS

RST(n)

This routine is needed to ensure correct handling of input/output to the conversational terminal. The convention is that writing is done to TAPE6 and reading comes from TAPE5, although these files are equated in the main program header. This routine is called when switching from a write to a read with n=6, and end of file mark is written. This ensures that all characters are sent to the terminal and the buffer is empty. After a read, rewinding tape 5 resets the buffer. It might be possible to convert all I/O operations to the UT CDC routines called IOP, which handle this problem automatically.

INUM(n)

This routine reads a line from the terminal and tries to turn it into an integer. If it cannot recognize all characters as either a blank or an integer, it will ask the operator to reenter.

MAGSET(N)

This routine takes the input number, which is "magnitude" as stored in IEN(,5) in the form of a four-digit integer and turns it into an A10 format string containing time, magnitude, direction, and probability in abbreviated form, or a message if N is zero.

CREATE(i1 . . . i7)

This routine creates a new entry in IEN and increments the counter NIEN. It writes an error message and stops if NIEN exceeds the dimensions of IEN.

DBIO(n)

This routine is used exactly as used in SETUP; however, the data base read (n=1) function is the only one used here. It reads the ESD matrix data base into the NAME and ND matrices. It also reads and displays the data base title.

PROGRAM LISTING

```
      PROGRAM EVAL(TAPE8,TAPE3,TTY=1002,TAPE5=TTY,TAPE6=TTY)
C ESD MATRIX FILE, CASE FILE, SYSTEM OUTPUT, OPERATOR INPUT=5,PRINT=6
C EVAL USES ECOLOGICAL SYYSSTEM FILES CREATED BY SETUP
C TO ASSIST OPERATOR IN EVALUATION OF A SINGLE SUB-ACTIVITY
C IN A SINGLE ECOSYSTEM
C ********
      COMMON/ESD/NNAME,NPEA,NAME(90,4),ND(90,50),NTITLE(8)
C  NNAME= NO. OF NAMES OF ATTRIBUTES
C  NPEA= NO. OF PEA NAMES
C  NAME( ) = NAME STORAGE
C  ND( )   = ESD MATRIX CREATED BY SETUP
C  NTITLE  = ESD SYSTEM NAME
      COMMON/EXEC/NIEN,NICOM,LEVEL,KSTATUS,ICTRL(10)
C NIEN=NUMBER OF IEN ENTRIES, NICOM= NUMBER OF COMMENTS
C LEVEL=CURRENT LEVEL OF ANALYS  KSTATUS=RECOVER STATUS
C ICTRL=MISC. CONTROL VARIABLES USED IN STOP/RESTART
      COMMON/ / IEN(400,7),ICOM(400,9),INDEX(400)
```

```
C IEN CONTAINS INDIVIDUAL DECISION ENTRIES AS FOLLOWS
C IEN( ,1)=LEVEL
C       2 =SERIAL NO. OF ATTRIBUTE OR PEA ENTRY WHICH LEAD TO THIS ENTRY
C        OR 9999 IF A COLLECTED ENTRY LED TO THIS ONE
C       3 =SERIAL NO. OF COLLECTED ENTRY WHICH THIS ENTRY IS COMBINED WITH
C       4 =ATTRIBUTE OR PEA NUMBER
C       5 = MAGNITUDE 4 DIGITS TIME/MAG/PROB/DIR
C       6 = COMMENT POINTER TO ICOM ARRAY
C       7 = STATUS OF ENTRY 0=DO NOT FOLLOW
C           1=MUST FOLLOW     2= ALREADY FOLLOWED
C           3 = THIS ENTRY COLLECTED INTO ANOTHER
C **********
C PROGRAM DESIGNER W.B. BROGDEN, JUNE, 1978
C
C*****
      INTEGER ANS(8),BLANK
      DATA BLANK/10H
C INITIALIZE ARRAYS
      DO 10 J=1,400
      INDEX(J)=0
      DO 6 K=1,7
    6 IEN(J,K)=0
      DO 10 K=1,9
   10 ICOM(J,K)=0
C INITIALIZE COUNTERS
      NIEN=0
      NICOM=0
      KSTATUS=0
C READ IN ESD MATRIX
      CALL DBIO(1)
C DETERMINE IF NEW ANALYSIS OR CONTINUATION OF OLD CASE
   40 WRITE(6,41)
   41 FORMAT(*NEW OR OLD ANALYSIS ?*)
      CALL RST(6)
      READ(5,43) ANS(1)
   43 FORMAT(8A10)
      CALL RST(5)
      IF(ANS(1).EQ.3HNEW)GOTO 100
      IF(ANS(1).EQ.3HOLD)GOTO 60
      GOTO 40
C HERE TO RECOVER OLD ANALYSIS FROM TAPE3 CASE FILE


   60 REWIND 3
      READ(3,61)NIEN,NICOM,LEVEL,KSTATUS,(ICTRL(J),I=1,10)
   61 FORMAT(14I5)
      IF(EOF,3) 64,68
   64 WRITE(6,65)
   65 FORMAT(*NO CASE FILE FOUND-PLEASE CHECK AGAIN*)
      STOP
```

```
   68 DO 70 J=1,NIEN
   70 READ(3,71)(IEN(J,K),K=1,7)
   71 FORMAT(15I5)
      DO 74 J=1,NICOM
   74 READ(3,75)(ICOM(J,K),K=1,9)
   75 FORMAT(I7,3X,8A10)
      READ(3,71)(INDEX(J),J=1,NIEN)
C FINISHED READING CASE FILE
      WRITE(5,85) NIEN,NICOM,LEVEL
   85 FORMAT(* CASE FILE READ, ENTRIES= *,I4,* COMMENTS=*,I4,*LEVEL=*,
     1 I4 )
      WRITE(6,87)(ICOM(1,K),K=2,9)
   87 FORMAT(* TITLE OF CASE *,/,8A10)
      GOTO(200,300,400,90),KSTATUS
   90 WRITE(6,91)
   91 FORMAT(* THIS CASE FILE FINISHED *)
      STOP
C HERE WITH NEW ANALYSIS
  100 WRITE(6,101)
  101 FORMAT(* NEW ANALYSIS, ENTER TITLE*)
      READ(5,103) (ICOM(1,J),J=2,9)
  103 FORMAT(8A10)
      CALL RST(5)
      WRITE(6,105)(ICOM(1,J),J=2,9)
  105 FORMAT(* TITLE *,/,8A10,/* OK? YES/NO ?*)
      CALL RST(6)
      READ(5,103) ANS(1)
      CALL RST(5)
      IF(ANS(1).EQ.3HYES) GOTO110
      GOTO100
C HERE WITH GOOD TITLE
  110 NICOM=1
C TITLE WAS ICOM(1   ) ALL FUTURE COMMENTS WILL BE ADDED THROUGH
C COMIN SUBROUTINE WHICH RETURNS THE NUMBER OF LINES ADDED OR 0
C
      CALL COMIN(1,N,ISTOP)
      IF(N.EQ.0) GOTO120
C HERE IF COMMENTS ADDED, LINK TO TITLE NEEDED
      ICOM(1,1)=2
  120 CALL PEA
C TO GET PEA-ATTRIBUTE LINKS AND SET UP INITIAL ENTRIES
C
      KSTATUS=0
C EST1 GENERATES BOTH SHORT AND LONG ENTRIES FROM THE PEAS
C RE-ENTRY FROM OLD ANALYSIS STARTS HERE
  200 CALL EST1
      KSTATUS=0
C START OF MAJOR LOOP FOR ALL LEVELS ABOVE THE FIRST
      LEVEL=1
C**************************
C
```

```
  300 CALL SUMSTOP
      KSTATUS=0
C SUMSTOP SUMMARIZES PREVIOUS ATTRIBUTE CHANGES AND GIVE CHANCE TO STOP
C
C NOW GET INTERACTION POSSIBILITIES FROM FSD MATRIX
      NOLD=NIEN
      CALL INTRACT(9,LEVEL)
C ENTRIES WILL HAVE STATUS=9, FOR NOT EVALUATED YET
      LEVEL=LEVEL+1
      IF(NOLD.EQ.NIEN)GOTO500
C HERE IF SOME NEW ENTRIES WERE CREATED
C NOW GET OPERATORS EVALUATION
  400 CALL EST2
      KSTATUS=0
      GOTO 300
C MAJOR CYCLE GOES BACK TO SUMSTOP
C ***********
C HERE ONLY IF NO NEW ENTRIES WERE CREATED
  500 KSTATUS=4
      WRITE(6,501) LEVEL
  501 FORMAT(* NO NEW ENTRIES AT LEVEL *I3,/* ANALYSIS STOPS*)
      CALL STOPSTA
      STOP
      END
C UTILITY SUBROUTINES FOR SETUP AND OTHER PROGRAMS




      SUBROUTINE RST(N)
C THIS ROUTINE IS NEEDED TO HANDLE INTERACTIVE I/O
      IF(N.EQ.6) END FILE 6
      REWIND N
      RETURN
      END
C
```

```
      SUBROUTINE INUM(I)
C TO GET A SINGLE INTEGER FROM INPUT ^ REWIND INPUT
      INTEGER A(10),NC(10)
      DATA (NC(K),K=1,10)/1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,
     1 1H8,1H9 /
    4 READ(5,7)(A(K),K=1,10)
    7 FORMAT(10A1)
      I=0
      DO 20 K=1,10
      IF(A(K).EQ.1H )GOTO20
C SKIP BLANKS
      DO 12 N=1,10
      IF(A(K).EQ.NC(N))GOTO14
   12 CONTINUE
C ERROR IF HERE, NON-NUMERIC
      WRITE(6,13) (A(J),J=1,10)
   13 FORMAT(* NON-NUMERIC IN *,10A1,* RE-ENTER*)
      CALL RST(6)
      GOTO4
   14 I=(I*10)+N-1
   20 CONTINUE
      RETURN
      END
C
C


      SUBROUTINE DBIO(N)
C N=1 READ,  N=0 WRITE DATA BANK OUT TO TAPE8
      COMMON/ESD/NNAME,NPEA,NAME(90,4),ND(90,50),NTITLE(8),
      REWIND 8
      IF(N.EQ.1) GOTO 100
C WRITE
      WRITE(6,1)
    1 FORMAT(* ENTER MOD NUMBER, 10 CHAR MAX*)
      CALL RST(6)
      READ(5,5) NTITLE(8)
      CALL RST(5)
    4 WRITE(8,5)(NTITLE(J),J=1,8),NNAME,NPEA
    5 FORMAT(8A10,4I4)
      DO 10 J=1,90
   10 WRITE(8,11) (NAME(J,K),K=1,4),(ND(J,K),K=1,NNAME)
   11 FORMAT(4A10,5X,50I1)
      WRITE(6,13)
   13 FORMAT(* WRITE*)
      WRITE(6,15)(NTITLE(K),K=1,8),NNAME,NPEA
   15 FORMAT(* DATA BASE *,/,8A10,/,* WITH *,I4,* ATT, PEA=*,I4)
      WRITE(6,21)
   21 FORMAT(* REMEMBER TO SAVE NEW DATA BASE*)
      CALL RST(6)
      RETURN
```

```
C
C  HERE TO READ
C
  100 READ(8,5) (NTITLE(J),J=1,8),NNAME,NPEA
      IF(EOF,8)102,108
  102 WRITE(6,103)
  103 FORMAT(* INVALID, NO DATA BASE*)
      STOP1
  108 DO 110 J=1,90
  110 READ(8,11)(NAME(J,K),K=1,4),(ND(J,K),K=1,NNAME)
      WRITE(6,113)
  113 FORMAT(* READ*)
      WRITE(6,15)(NTITLE(K),K=1,8),NNAME,NPEA
      CALL RST(6)
      RETURN
      END
C
C


      SUBROUTINE COMIN(LINK,NCNT,ISTOP)
C GETS USER COMMENTS, LINK FIRST COMMENTS TO IEN, SUBSEQUENT TO ICOM
C RETURN NCNT=NUMBER OF COMMENTS ADDED
      COMMON/EXEC/NIEN,NICOM,LEVEL,KSTATUS,ICTRL(10)
      COMMON/ / IEN(400,7),ICOM(400,9),INDEX(400)
      INTEGER ANS(8)
   10 WRITE(6,11)
   11 FORMAT(* ENTER COMMENTS, TERMINATE WITH BLANK LINE *)
      CALL RST(6)
      ISTOP=0
      NCNT=0
C NCNT IS COUNT OF LINES ENTERED
   15 READ(5,17)(ANS(J),J=1,8)
   17 FORMAT(8A10)
      CALL RST(5)
C CHECK FOR BLANK LINE OR STOP
      IF(ANS(1).EQ.10H              )RETURN
      IF(ANS(1).EQ.5H$STOP)GOTO100
C INSERT COMMENT
      NICOM=NICOM+1
      NCNT=NCNT+1
      DO 30 J=2,9
   30 ICOM(NICOM,J)=ANS(J-1)
C LINK IEN OR ICOM TO CURRENT COMMENT
      IF(NCNT.EQ.1) IEN(LINK,6)=NICOM
      IF(NCNT.GT.1) ICOM(NIOLD,1)=NICOM
      NIOLD=NICOM
      GOTO15
  100 ISTOP=1
      RETURN
      END
```

```
      SUBROUTINE PEA
      COMMON/ESD/NNAME,NPEA,NAME(90,4),ND(90,50),NTITLE(8)
C THIS SUBROUTINE GOES THROUGHT THE LIST OF POTENTIAL PEA#S IN THE ND MATRIX
C ASKING THE OPERATOR IF THE PEA OCCURS
      COMMON/EXEC/NIEN,NICOM,LEVEL,KSTATUS,ICTRL(10)
      COMMON/ / IEN(400,7),ICOM(400,9),INDEX(400)
C**********************
      WRITE(6,11) NPEA
   11 FORMAT(I4,* POSSIBLE PEAS IN THIS ECOSYSTEM *)
      DO 100 N=1,NPEA
      JP=91-N
C PEA LIST RUNS UP FROM BOTTOM OF ND MATRIX
   20 WRITE(6,21)(NAME(JP,K),K=1,4)
   21 FORMAT(*DOES *,4A10,* OCCURS YES/NO*)
      CALL RST(6)
      READ(5,23) NANS
   23 FORMAT(A10)
      IF(NANS.EQ.3HYES)GOTO30
      IF(NANS.EQ.2HNO) GOTO100
      GOTO20
C HERE WITH PEA OCCURING, GENERATE ENTRY
   30 CALL CREATE(0,0,0,JP,0,0,1)
C STATUS = MUST FOLLOW
C OTHERS ARE PRESET=0
  100 CONTINUE
      WRITE(6,103)NIEN
  103 FORMAT(I4,* INITIAL PEAS*)
      RETURN
      END


      SUBROUTINE EST1
C GETS USER INPUT FROM LEVEL 1 CHANGES ONLY
C WITH STATUS SET=1, MUST FOLLOW
      COMMON/ESD/NNAME,NPEA,NAME(90,4),ND(90,50),NTITLE(8)
      COMMON/EXEC/NIEN,NICOM,LEVEL,KSTATUS,ICTRL(10)
      INTEGER ANS
      COMMON/ / IEN(400,7),ICOM(400,9),INDEX(400)
C**********************
      IF(KSTATUS.EQ.0)GOTO100
C HERE RESTART, RECALL LOOP PARAMETERS
   10 JTPEA=ICTRL(1)
      LPEA =ICTRL(2)
      JA   =ICTRL(3)
      GOTO200
C HERE NORMAL ENTRY, SET UP LOOPS
C ALL ENTRIES ARE PEAS AT THIS POINT
  100 LPEA=NIEN
      JTPEA=1
      JA=1
```

```
C LOOP TO HERE FOR PEAS
  200 JP=IEN(JIPEA,4)
C MODS AUG-6,78 TO IMPROVE $STOP
C JP GIVES INDEX TO NAME AND ND ARRAY, JIPEA JUST COUNTS
C LOOP CONTROL FOR ATTRIBUTES IN ND MATRIX
  300 IF(JA.LE.NNAME)GOTO318
C FINISHED PEA, MARK STATUS=FOLLOWED
      IEN(JIPEA,7)=2
      JA=1
      JIPEA=JIPEA+1
      IF(JIPEA.GT.LPEA)RETURN
C RECYCLE FOR MORE PEAS
      GOTO200
C DOES THIS PEA AFFECT THIS ATTRIBUTE?
  318 IF(ND(JP,JA).NE.0)GOTO320
  319 JA=JA+1
      GOTO300
C*************************
C GET OPERATOR OPINION ON SIGNIFICANCE
  320 WRITE(6,321)(NAME(JA,K),K=1,4),(NAME(JP,K),K=1,4)
  321 FORMAT(* IS *,4A10,/* SIGNIFICANTLY AFFECTED BY PEA *,4A10)
      CALL RST(6)
      READ(5,323)ANS
  323 FORMAT(A10)
      CALL RST(5)
      IF(ANS.EQ.5H$STOP)GOTO400
      IF(ANS.EQ.3HYES)  GOTO500
      IF(ANS.EQ.2HNO)   GOTO319
      WRITE(6,325)
  325 FORMAT(* POSSIBLE ANSWERS= YES  NO  $STOP *)
      GOTO320
C*************************
C TO STOP, TRANSFER LOOP PARAMETERS TO ICTRL
  400 ICTRL(1)=JIPEA
      ICTRL(2)=LPEA
      ICTRL(3)=JA
      KSTATUS=1
      CALL STOPSTA
C IF RETURN FROM STOPSTA, CONTINUE


      GOTO10
C*************************
C GENERATE SHORT ENTRY
  500 CALL CREATE(1,JIPEA,0,JA,2000,0,9)
      WRITE(6,511)
  511 FORMAT(* SHORT TERM EFFECT*)
      CALL ENTEST(NIEN,ISTOP)
      IF(ISTOP.GT.0)GOTO550
```

```
C GENERATE LONG ENTRY
      CALL CREATE(1,JIPEA,0,JA,1000,0,9)
      WRITE(6,521)
  521 FORMAT(* LONG TERM EFFECT*)
      CALL ENTEST(NIEN,ISTOP)
      IF(ISTOP.GT.0)GOTO550
C GO BACK FOR MORE ATTRIBUTES
      GOTO319
C*********************
C HERE TO STOP WITH LAST ENTRY PROBABLY INVALID
  550 NIEN=NIEN-1
      GOTO400
      END


      SUBROUTINE SUMSTOP
C SUMMARIZE PROGRESS, PERMIT STOPPING, AND PERMIT MODIFICATION
C TO EARLIER ENTRIES, WITH SUBSEQUENT TRIMING OF RESULTANT TREE
      COMMON/EXEC/NIEN,NICOM,LEVEL,KSTATUS,ICTRL(10)
      COMMON/ / IEN(400,7),ICOM(400,9),INDEX(400)
       INTEGER ANS,LHIST(10)
C*****************
C INITIALIZE HISTOGRAM COUNTERS
      DO 10 J=1,10
   10 LHIST(J)=0
C COUNT ENTRIES/LEVELS
      DO 20 J=1,NIEN
      JP=IEN(J,1)+1
   20 LHIST(JP)=LHIST(JP)+1
      WRITE(6,25)
   25 FORMAT(* SUMMARY OF PROGRESS SO FAR*)
      WRITE(6,29) NIEN,NICOM
   29 FORMAT(I4,* ENTRIES, COMMENTS= *,I4)
      DO 40 J=0,LEVEL
   40 WRITE(6,41) J,LHIST(J+1)
   41 FORMAT(*LEVEL*,I3,I5,* ENTRIES*)
  100 WRITE(6,101)
  101 FORMAT(* PRESENT OPTIONS *,/
     1 * 0 - CONTINUE ANALYSIS*,/
     2 * 1 - OUTPUT SUMMARY OF LAST LEVEL*,/
     3 * 2 - $STOP - SAVE CURRENT CASE FILE*,/
     4 * 3 - TRACE AND MODIFY*,/
     5 )
C*****
      CALL RST(6)
      CALL INUM(N)
      IF(N.GT.3) GOTO100
      IF(N.EQ.0) RETURN
      GOTO (200,300,400,500,600),N
C ***************)
  200 CALL SMRZE
C OUTPUT SUMMARY THIS LEVEL
      GOTO100
```

```
C**************
C STOP
  300 KSTATUS=2
      CALL STOPSTA
      GOTO 100
C**********
C TRACE AND ALTER EARLIER
  400 CALL TRACE
      GOTO 100
C**********
C DUMMY STATEMENTS
  500 CONTINUE
  600 CONTINUE
  700 STOP1
      END

      SUBROUTINE INTRACT(NST,LEV)
C SEARCH ALL IEN FOR ENTRIES OF LEVEL LEV AND STATUS=1 (MUST FOLLOW)
C AND GENERATE NEXT HIGHER LEVEL ENTRIES WITH STATUS SET=NST
      COMMON/ESD/NNAME,NPEA,NAME(90,4),ND(90,50),NTITLE(8)
      COMMON/EXEC/NIEN,NICOM,LEVEL,KSTATUS,ICTRL(10)
      COMMON/ / IEN(400,7),ICOM(400,9),INDEX(400)
C
C******
      NIENT=NIEN
      NCT=0
C NCT=COUNT OF NEW ENTRIES GENERATED
      DO 100 I=1,NIENT
      IF(IEN(I,1).NE.LEV) GOTO100
      IF(IEN(I,7).NE.1)GOTO100
C STATUS=1 - MUST FOLLOW
C HERE WITH ENTRY
      N=IEN(I,4)
C N=NAME INDEX
      DO 60 J=1,NNAME
C LOOK FOR INTERACTING ATTRIBUTES
      IF(ND(N,J).EQ.0)GOTO60
C HERE WITH INTERACTION FOUND
C CREATE NEW ENTRY
      CALL CREATE(LEV+1,I,0,J,0,0,NST)
C SET LEVEL,PRECEDING SERIAL NO, NAME, STATUS
      NCT=NCT+1
   60 CONTINUE
C SET STATUS OF GENERATING ENTRY=2 =FOLLOWED=
      IEN(I,7)=2
  100 CONTINUE
C TEMP MESSAGE
      WRITE(6,111)NCT,LEV,NST
  111 FORMAT(I4,* NEW ENTRIES GENERATED DUE TO LEVEL*,I4,* STAT=*
     1 I4 )
      RETURN
      END
```

```
      SUBROUTINE ENTEST(JENT,ISTOP)
C A ROUTINE TO OVERSEE AQUISITION OF MAGNITUDE ESTIMATE FOR
C ENTRY JENT,   ISTOP IS USED TO STOP ANALYSIS ABNORMALLY
C*****************
      COMMON/ESD/NNAME,NPEA,NAME(90,4),ND(90,50),NTITLE(8)
      COMMON/EXEC/NIEN,NICOM,LEVEL,KSTATUS,ICTRL(10)
      COMMON/ / IEN(400,7),ICOM(400,9),INDEX(400)
      INTEGER ANS
C*****************
C FIRST DISPLAY INTERACTION
      JNAME=IEN(JENT,4)
      JCAUSE=IEN(JENT,2)
      JLEV=IEN(JENT,1)
      JMAG=0
      IF(JCAUSE.LT.999)JMAG=IEN(JCAUSE,5)
      CALL MAGSET(JMAG)
C JMAG IS A10 MAGNITUDE
      JCNAM=0
      IF(JCAUSE.LT.999)JCNAM=IEN(JCAUSE,4)
C CAUSE SERIAL NUMBER MAY BE 9999 TO INDICATE A COLLECTED ENTRY
C OTHERWISE JCNAM IS THE NAME OF THE ATTRIBUTE IN THE CAUSE ENTRY
C
  100 WRITE(6,101)(NAME(JNAME,K),K=1,4),JLEV
  101 FORMAT(* EVALUATE CHANGE IN *,4A10,* LEVEL *,I2,* DUE TO*)
      IF(JCNAM.EQ.0)WRITE(6,103)
  103 FORMAT(* TOTAL EFFECTS AT THIS LEVEL *)
      IF(JCNAM.GT.0)WRITE(6,105)(NAME(JCNAM,K),K=1,4),JMAG
  105 FORMAT(2X,4A10,* = *,A10)
C
      CALL GETMC(ISTOP,JENT,IMAG)
C CHECK ISTOP, THEN FILL MAGNITUDE
      IF(ISTOP.NE.0) RETURN
      IEN(JENT,5)=IMAG
C NOW DETERMINE IF ENTRY IS TO BE FOLLOWED
  200 WRITE(6,201)
  201 FORMAT(* SHOULD THIS ENTRY BE FOLLOWED *)
      CALL RST(6)
  210 READ(5,211) ANS
      CALL RST(5)
  211 FORMAT(A10)
      CALL RST(5)
      IF(ANS.EQ.3HYES)GOTO220
      IF(ANS.EQ.2HNO)GOTO230
      IF(ANS.EQ.5H$STOP)ISTOP=1
      IF(ANS.EQ.5H$STOP)RETURN
      WRITE(6,215)ANS
  215 FORMAT(A10,* NOT RECOGNIZED, YES OR NO*)
      GOTO200
```

```
C
  220 IEN(JENT,7)=1
C MUST FOLLOW
      RETURN
  230 IEN(JENT,7)=0
C DO NOT FOLLOW
      RETURN
      END


      SUBROUTINE MAGSET(JMAG)
C RETURN A10 FORMAT MAGNITUDE FOR ENTRY JCAUSE
C OR BLANK IF JCAUSE.GT.999 (COLLECTED ENTRY)
C FOLLOWING ARE DEFINED IN GETMC
      COMMON/MAGWDS/IT(3),IM(10),ID(3),IP(4)
C SEPARATE INDEXES
      IF(JMAG.EQ.0)GOTO20
      J3=JMAG/1000
      J2=(JMAG-(J3*1000))/100
      J1=(JMAG-(J3*1000)-(J2*100))/10
      J0=JMAG-(J3*1000)-(J2*100)-(J1*10)
      ENCODE(10,15,JMAG)IT(J3+1),IM(J2+1),ID(J1+1),IP(J0+1)
   15 FORMAT(A3,A3,A2,A2)
      RETURN
   20 JMAG=10HNOT EVAL.
      RETURN
      END


      SUBROUTINE GETMC(ISTOP,LINK,IMAG)
C  ISTOP USED TO SIGNAL INPUT OF $STOP COMMAND, ISTOP=1 STOP, 0 OK
C  LINK IS POINTER TO IEN ENTRY
C  MAG IS RETURNED WITH 4 DIGIT MAGNITUDE, ETC. CODE
      COMMON/EXEC/NIEN,NICOM,LEVEL,KSTATUS,ICTRL(10)
      COMMON/ / IEN(400,7),ICOM(400,9),INDEX(400)
C
      INTEGER TIME,MAG,DIR,PROB
      COMMON/MAGWDS/TIME(3),MAG(10),DIR(3),PROB(4)
C SYMBOLS TO RECOGNIZE
      DATA (TIME(J),J=1,3)/3H    ,3HLO ,3HSH  /
       DATA(MAG(J),J=1,5)/3H    ,3HM9 ,3HM8 ,3HM7 ,3HM6  /
      DATA(MAG(J),J=6,10)/3HM5 ,3HM4 ,3HM3 ,3HM2 ,3HM1  /
      DATA (DIR(J),J=1,3)/3H    ,3HI  ,3HO   /
      DATA (PROB(J),J=1,4)/3H     ,3HDF ,3HPR ,3HPO  /
C NOTE ORDER GIVES SORT PRIORITY WITH 1 FIRST, ETC.
```

```
C
      INTEGER LN(20),ISL(4),ANS(4)
      ISTOP=0
    2 WRITE(6,3)
    3 FORMAT(* ENTER EFFECT ESTIMATE*)
      CALL RST(6)
      READ(5,5)(LN(K),K=1,20)
    5 FORMAT(20A1)
      CALL RST(5)
C CHECK FOR $STOP
      IF(LN(1).EQ.1H$)GOTO1000
C NOW LOCATE SLASHES. FORM SHOULD BE TIME/MAG/DIR/PROB
      ICSL=0
      DO 10 J=1,20
      IF(LN(J).NE.1H/)GOTO10
      ICSL=ICSL+1
      ISL(ICSL)=J
   10 CONTINUE
C IF NOT 3 SLASHES, ERROR
      IF(ICSL.NE.3)GOTO100
C SET LAST MARKER
      ISL(4)=ISL(3)+3
C NOW ENCODE FROM LN INTO ANS
      J=1
      DO 20 K=1,4
      N=ISL(K)-J
      IS=ISL(K)-1
      ENCODE(N,21,ANS(K))(LN(L),L=J,IS)
   20 J=ISL(K)+1
   21 FORMAT(10A1)
C NOW TRY TO MATCH
C CHECK FOR VALIDITY, TIME FIRST
      DO 40 J=1,3
      IF(ANS(1).EQ.TIME(J)) GOTO45
   40 CONTINUE
      GOTO100
   45 NTIME=J-1
C CHECK MAGNITUDE
      DO 50 J=1,10
      IF(ANS(2).EQ.MAG(J)) GOTO55
   50 CONTINUE
      GOTO100

   55 NMAG=J-1
C CHECK DIRECTION
      DO 60 J=1,3
      IF(ANS(3).EQ.DIR(J))GOTO65
   60 CONTINUE
      GOTO100
   65 NDIR=J-1
```

```
C CHECK PROBABILITY
      DO 70 J=1,4
      IF(ANS(4).EQ.PROB(J)) GOTO75
   70 CONTINUE
      GOTO100
   75 NPROB=J-1
      GOTO120
C HERE TO WRITE OUT VALID ENTRIES
  100 WRITE(6,101)(TIME(J),J=1,3)
  101 FORMAT(* INPUT FORM IS= TIME/MAG/DIR/PROB */* TIME CAN BE *,
     1 3A3)
      WRITE(6,103)(MAG(J),J=1,10)
  103 FORMAT(* MAGNITUDE *,10A3)
      WRITE(6,105)(DIR(J),J=1,3)
  105 FORMAT(* DIRECTION *,3A3)
      WRITE(6,107)(PROB(J),J=1,4)
  107 FORMAT(* PROBABILITY *,4A3)
      GOTO2
C HERE WITH VALID MAGNITUDE ESTIMATE
  120 IMAG=(1000*NTIME)+(100*NMAG)+(10*NDIR)+NPROB
      CALL COMIN(LINK,NCNT,ISTOP)
C ISTOP=1 IF $STOP ENTERED
      RETURN
C**********
C HERE IF $STOP ENTERED
C**********
 1000 ISTOP=1
      RETURN
C CALLING PROGRAM WILL THEN CALL STOPSTA WITH APPROPRIATE PARAMETERS
      END



      SUBROUTINE STOPSTA
C STORE CASE FILE WITH KSTATUS INDICATING CALLING SUBROUTINE
C GIVES OPERATOR OPTION OF STOPING OR CONTINUEING
      COMMON/EXEC/NIEN,NICOM,LEVEL,KSTATUS,ICTRL(10)
      COMMON/ / IEN(400,7),ICOM(400,9),INDEX(400)
      INTEGER ANS
C
C**********
C
      REWIND 3
C WRITE MAIN CONTROL VARIABLES
      WRITE(3,21)NIEN,NICOM,LEVEL,KSTATUS,(ICTRL(J),J=1,10)
   21 FORMAT(14I5)
      DO 30 J=1,NIEN
   30 WRITE(3,31)(IEN(J,K),K=1,7)
   31 FORMAT(9I5)
      DO 40 J=1,NICOM
   40 WRITE(3,41)(ICOM(J,K),K=1,9)
```

```
   41 FORMAT(I7,3X,8A10)
      WRITE(3,51)(INDEX(J),J=1,NIEN)
   51 FORMAT(15I5)
C ***********
      WRITE(6,61)NIEN,NICOM
   61 FORMAT(* CASE FILE WRITTEN. ENTRIES=*,I4,* COMMENTS= *,I4)
      WRITE(6,63)(ICOM(1,J),J=2,9)
   63 FORMAT(* TITLE= *,8A10)
      WRITE(6,65) LEVEL,KSTATUS
   65 FORMAT(* LEVEL=*,I4,* STATUS=*,I4)
   70 WRITE(6,71)
   71 FORMAT(* DO YOU WANT TO CONTINUE WITH ANALYSIS ? YES/NO*)
      CALL RST(6)
      READ(5,73) ANS
   73 FORMAT(A10)
      CALL RST(5)
      IF(ANS.EQ.2HNO) STOP
      IF(ANS.EQ.3HYES) RETURN
      GOTO70
      END


      SUBROUTINE EST2
C A ROUTINE TO GET MAGNITUDE ESTIMATES AND CONTROL COLLECTION
C OF CHANGES , DISPLAY OF PREVIOUS CHANGES, ETC
C***********
      COMMON/ESD/NNAME,NREA,NAME(90,4),ND(90,50),NTITLE(8)
      COMMON/EXEC/NIEN,NICOM,LEVEL,KSTATUS,ICTRL(10)
      COMMON/ / IEN(400,7),ICOM(400,9),INDEX(400)
      INTEGER ANS
C*************
C CHECK IF A RE-START
      IF(KSTATUS.GT.0)GOTO100
   10 NA=0
      NIENOLD=NIEN
      CALL SORT(4,0)
C SORT INDEX ARRAY BY ATTRIBUTE NUMBER, NEW START
      GOTO200
C RESTART USES OLD INDEX ARRAY TO AVOID CONFUSION WITH NEWLY CREATED
C COLLECTED ENTRIES
C RESTART
  100 NA=ICTRL(1)
      NIENOLD=ICTRL(2)
C NOTE - NIENOLD IS NEEDED BECAUSE COLLECTION OF ENTRIES WILL CREATE
C NEW ENTRIES DURING RUNNING OF EST2, THUS FOULING UP THE -220 LOOP
C*************
C LOOP CONTROL, FOR ALL ENTRIES
  200 NA=NA+1
      IF(NA.GT.NNAME) RETURN
      ISTART=1
      ILAST=0
      DO 220 J=1,NIENOLD
```

```
C FOR ALL ENTRIES, LOCATE START AND LAST OF IEN FOR NA
      JTEMP=INDEX(J)
      JNAME=IEN(JTEMP,4)
      IF(JNAME.LT.NA)ISTART=J+1
      IF(JNAME.EQ.NA)ILAST=J
  220 CONTINUE
C ANY ENTRIES THIS ATTRIBUTE>
      IF(ILAST.EQ.0)GOTO200
C COUNT ENTRIES WITH NA WHICH HAVE NOT BEEN EVALUATED AT THIS LEVEL
      JCT=JSHORT=JLONG=0
C JCT IS COUNT OF ENTRIES AT THIS LEVEL NOT EVAL. YET
C JSHORT & JLONG COUNT CAUSES OF ALL ENTRIES AT THIS LEVEL. NA
      DO 230 J=ISTART,ILAST
      JT=INDEX(J)
      IF(IEN(JT,1).NE.LEVEL)GOTO230
      JCAUSE=IEN(JT,2)
      IF(IEN(JCAUSE,5).GT.1999) JSHORT=JSHORT+1
      IF(IEN(JCAUSE,5).LT.1999) JLONG=JLONG+1
      IF(IEN(JT,7).EQ.9)JCT=JCT+1
  230 CONTINUE
      IF(JCT.EQ.0)GOTO200
C**********************
C HERE WITH AT LEAST ONE UNEVALUATED CHANGE IN ATTRIBUTE NA
C IN ENTRIES POINTED TO BY INDES(ISTART) -INDEX(ILAST)
  240 WRITE(6,241)(NAME(NA,K),K=1,4)
  241 FORMAT(* PREVIOUS CHANGES AFFECTING *,4A10)
      DO 250 J=ISTART,ILAST
      JT=INDEX(J)


      IF(IEN(JT,1).LT.LEVEL) CALL ATOUT(JT)
  250 CONTINUE
C NOW LOOK AT PRESENT LEVEL
      WRITE(6,255)(NAME(NA,K),K=1,4)
  255 FORMAT(* AT PRESENT LEVEL *,4A10,* IS CHANGED BY*)
      CALL RST(6)
      DO 260 J=ISTART,ILAST
      JT=INDEX(J)
      IF(IEN(JT,1).EQ.LEVEL) CALL ATOUT(JT)
  260 CONTINUE
C NOW GET EVALUATION FOR ALL LEVEL=PRESENT AND NOT EVAL
      DO 280 J=ISTART,ILAST
      JT=INDEX(J)
      IF((IEN(JT,1).NE.LEVEL).OR.(IEN(JT,7).NE.9)) GOTO280
      CALL ENTEST(JT,ISTOP)
      IF(ISTOP.GT.0)GOTO600
  280 CONTINUE
C**********************
C FINISHED WITH ESTIMATES FOR THIS ATTRIBUTE
```

```
C IF MORE THAN ONE ENTRY, A COMBINED ENTRY CAN BE GENERATED
  300 IF(JCT.LE.1)GOTO200
      IF((JSHORT.LE.1).AND.(JLONG.LE.1))GOTO200
  302 WRITE(6,303)JCT,(NAME(NA,K),K=1,4)
  303 FORMAT(*DO YOU WISH TO COMBINE THE EFFECTS OF THE *,I4* ENTRIES*/
     1 * AFFECTING *,4A10)
      CALL RST(6)
  310 READ(5,311)ANS
  311 FORMAT(A10)
      CALL RST(5)
      IF(ANS.EQ.2HNO)GOTO200
      IF(ANS.EQ.5H$STOP)GOTO600
      IF(ANS.EQ.3HYES)GOTO320
      WRITE(6,313)
  313 FORMAT(* YES OR NO OR $STOP*)
      CALL RST(6)
      GOTO310
C**********
C HERE TO CONSOLIDATE ENTRIES
C CHECK SHORT FIRST
  320 IF(JSHORT.LE.1)GOTO500
  400 WRITE(6,401)
  401 FORMAT(* FOR SHORT TIME PERIOD2 YES/NO*)
      CALL RST(6)
      READ(5,311)ANS
      CALL RST(5)
      IF(ANS.EQ.3HYES)GOTO410
      IF(ANS.EQ.2HNO)GOTO500
      IF(ANS.EQ.5H$STOP)GOTO600
      GOTO400
  410 CALL CREATE(LEVEL,9999,0,NA,0,0,9)
C NOW LABEL ALL SHORT ENTRIES
      DO 420 J=ISTART,ILAST
      JT=INDEX(J)
      IF(IEN(JT,1).NE.LEVEL)GOTO420
      IF(IEN(JT,5).LT.1999)GOTO420
C REMAINDER ARE SHORT AT THIS LEVEL AND THIS ATTRIBUTE
      IEN(JT,3)=NIEN
      IEN(JT,7)=3
  420 CONTINUE
      CALL ENTEST(NIEN,ISTOP)
      IF(ISTOP.GT.0)GOTO600
C**********
C NOW FOR LONG ENTRIES
  500 IF(JLONG.LE.1)GOTO200
  502 WRITE(6,503)
  503 FORMAT(* FOR LONG TIME PERIOD2 YES/NO *)
      CALL RST(6)
      READ(5,311)ANS
```

```
      CALL RST(5)
      IF(ANS.EQ.3HYES)GOTO510
      IF(ANS.EQ.2HNO) GOTO200
      IF(ANS.EQ.5H$STOP)GOTO600
      GOTO502
  510 CALL CREATE(LEVEL,9999,0,NA,0,0,9)
C NOW LABEL ALL LONG ENTRIES
      DO 520 J=ISTART,ILAST
      JT=INDEX(J)
      IF(IEN(JT,1).NE.LEVEL)GOTO520
      IF(IEN(JT,5).GT.1999)GOTO520
C HERE WITH LONG AT THIS LEVEL
      IEN(JT,3)=NIEN
      IEN(JT,7)=3
  520 CONTINUE
      CALL ENTEST(NIEN,ISTOP)
      IF(ISTOP.GT.0)GOTO600
      GOTO200
C**********
C HERE TO STOP
  600 KSTATUS=3
      NA=NA-1
C HAVE TO BACK UP ONE (SEE LINE 200)
      ICTRL(1)=NA
      ICTRL(2)=NIENOLD
      CALL STOPSTA
C IF IT RETURNS, CONTINUE
      GOTO200
      END



      SUBROUTINE ATOUT(JT)
C TO DISPLAY THE CAUSE OF ENTRY JT
      COMMON/EXEC/NIEN,NICOM,LEVEL,KSTATUS,ICTRL(10)
      COMMON/ESD/NNAME,NPEA,NAME(90,4),ND(90,50),NTITLE(8)
      COMMON/ / IEN(400,7),ICOM(400,9),INDEX(400)
C**************
      JC=IEN(JT,2)
      JMAG=IEN(JT,5)
      CALL MAGSET(JMAG)
      IF(JC.GT.9000) GOTO100
C HERE NOT A COLLECTED ENTRY
      JCNAM=IEN(JC,4)
      IF(JCNAM.GT.NNAME)GOTO60
C HERE IF NORMAL ATTRIBUTE
      JCMAG=IEN(JC,5)
      CALL MAGSET(JCMAG)
```

```
C READY TO PRINT NAME AND MAGNITUDE
      WRITE(6,21)(NAME(JCNAM,K),K=1,4),JCMAG,JMAG,JT
   21 FORMAT(4A10,X,A10,* CAUSES *,A10,I3)
      RETURN
C HERE PEA
   60 WRITE(6,61)(NAME(JCNAM,K),K=1,4),JMAG
   61 FORMAT(*PEA *,4A10,* CAUSES *,A10)
      RETURN
C HERE COLLECTED ENTRY
  100 WRITE(6,101) JMAG,TEN(JT,1)
  101 FORMAT(* COLLECTED EFFECTS AT PREVIOUS LEVEL CAUSES *,A10,* LEV*
     1 ,I3)
      RETURN
      END



      SUBROUTINE SORT(JI,ITYPE)
C  JI=COLUMN OF TEN MATRIX TO USE IN REORDERING INDEX
C IF ITYPE=0 SET INDEX=1,2,..NIEN  ELSE LEAVE ALONE
      COMMON/EXEC/NIEN,NICOM,LEVEL,KSTATUS,ICTRL(10)
      COMMON/ / IEN(400,7),ICOM(400,9),INDEX(400)
      INTEGER TEMP(400)
      IF(ITYPE.GT.0)GOTO40
C IF NOT 0 USE PREVIOUS ORDER IN LOADING TEMP (EXPERIMENTAL)
      DO 10 J=1,NIEN
   10 TEMP(J)=IEN(J,JI)*1000+J
C THIS VECTOR WILL BE SORT KEY
      DO 20 J=1,NIEN
   20 INDEX(J)=J
C INDEX VECTOR IS MOVED ALONG WITH KEY BY ISORT
   30 CALL ISORT(NIEN,TEMP(1),INDEX(1))
      RETURN
   40 DO 50 J=1,NIEN
      JT=INDEX(J)
   50 TEMP(J)=IEN(JT,JI)*1000+J
      GOTO30
      END
```

```
      SUBROUTINE CREATE(I1,I2,I3,I4,I5,I6,I7)
C CREATES NEW ENTRY IN IEN
      COMMON/EXEC/NIEN,NICOM,LEVEL,KSTATUS,ICTRL(10)
      COMMON/ / IEN(400,9),ICOM(400,9),INDEX(400)
      NIEN=NIEN+1
      IF(NIEN.LT.401)GOTO20
      WRITE(6,1)NIEN
    1 FORMAT(* ENTRY ARRAY LIMIT OF *,I4,* HAS BEEN EXCEEDED*)
      STOP
   20 IEN(NIEN,1)=I1
      IEN(NIEN,2)=I2
      IEN(NIEN,3)=I3
      IEN(NIEN,4)=I4
      IEN(NIEN,5)=I5
      IEN(NIEN,6)=I6
      IEN(NIEN,7)=I7
      RETURN
      END
```

```
      SUBROUTINE SMRZE
C TO SUMMARIZE RESULTS AT THE PRESENT LEVEL
      COMMON/EXEC/NIEN,NICOM,LEVEL,KSTATUS,ICTRL(10)
      COMMON/ESD/ NNAME,NPEA,NAME(90,4),ND(90,50),NTITLE(8)
      COMMON/ / IEN(400,7),ICOM(400,9),INDEX(400)
      INTEGER ANS
C***************
C $STOP CAUSES RETURN TO CALLING PROGRAM
C NOW SORT IEN BY MAGNITUDE CODES WHICH SETS ORDER OF OUTPUT
      CALL SORT(5,0)
C
   20 WRITE(6,21)
   21 FORMAT(* OPTIONS ARE*/* 1= ALL ATTRIBUTES *
     1 /* 2= INDIVIDUAL SELECTION*
     1 /* 0= CONTINUE ANALYSIS*
     2 )
      CALL RST(6)
      CALL INUM(ITYPE)
      IF(ITYPE.LE.0)RETURN
      IF(ITYPE.EQ.2)GOTO100
      IF(ITYPE.EQ.1)GOTO80
      GOTO20
```

```
C**************************
C LOOP CONTROL FOR ALL ATTRIBUTES
   80 N=0
   90 N=N+1
      IF(N.GT.NNAME)RETURN
      DO 94 J=1,NIEN
      IF(IEN(J,1).NE.LEVEL)GOTO94
      IF(IEN(J,4).EQ.N)GOTO110
C TO AVOID PRINTING HEADER FOR ATTRIBUTES WITH NO CHANGES
   94 CONTINUE
      GOTO90
  100 WRITE(6,101)
  101 FORMAT(* ENTER NUMBER OF ATTRIBUTE OR 0 TO QUIT*)
      CALL RST(6)
      CALL INUM(N)
      IF(N.LE.0)RETURN
      IF(N.LE.NNAME)GOTO110
      WRITE(6,103) NNAME
  103 FORMAT(*ATTRIBUTE NUMBERS FROM 1 -*,I2)
      GOTO100
  110 WRITE(6,111)(NAME(N,K),K=1,4),LEVEL
  111 FORMAT(*ATTRIBUTE *,4A10,/* IS ALTERED AT LEVEL*,I2,* BY*)
C LOOK THROUGHIEN IN ORDER OF SORT, SELECT ONLY *N* ALTERATIONS
      DO 140 J=1,NIEN
      JT=INDEX(J)
      IF(IEN(JT,4).NE.N)GOTO140
      IF(IEN(JT,1).NE.LEVEL)GOTO140
      CALL ATOUT(JT)
      JCOM=IEN(JT,6)
      IF(JCOM.EQ.0)GOTO140
      CALL COMOUT(JCOM,1)
C WRITE COMMENTS USING FORMAT 1
  140 CONTINUE
      IF(ITYPE.EQ.2)GOTO100
      IF(ITYPE.EQ.1)GOTO90
C ELSE ERROR



      STOP1
      END
```

```
      SUBROUTINE COMOUT(NC,NF)
C OUTPUT COMMENT NC AND ALL FOLLOWING COMMENTS IN FORMAT NF
      COMMON/EXEC/NIEN,NICOM,LEVEL,KSTATUS,ICTRL(10)
      COMMON/ / IEN(400,7),ICOM(400,9),INDEX(400)
C CHECK RANGE OF NC
   10 IF((NC.LE.0).OR.(NC.GT.NICOM))GOTO100
C CHECK FORMAT IN RANGE
      IF(NF.NE.1)GOTO100
C ADD MORE LATER
      WRITE(6,1)(ICOM(NC,K),K=2,9)
      NC=ICOM(NC,1)
      IF(NC.EQ.0)RETURN
C AFTER LAST CONNECTED COMMENT
      GOTO10
C NOW FORMATS
    1 FORMAT(8A10)
C ERROR
  100 WRITE(6,101)NC,NF
  101 FORMAT(* PROBLEM IN COMOUT, NC,NF=*,2I5)
      RETURN
      END



      SUBROUTINE TRACE
C TO TRACE THE EFFECTS OF REMOVING AN ENTRY
C
      COMMON/ESD/NNAME,NPEA,NAME(90,4),ND(90,50),NTITLE(8)
      COMMON/STAK/ISP,ISTKMX,ISTK(200)
      COMMON/EXEC/NIEN,NICOM,LEVEL,KSTATUS,ICTRL(10)
      COMMON/ / IEN(400,7),ICOM(400,9),INDEX(400)
C**********************
C FIRST GET NUMBER OF CAUSITIVE ENTRY
   10 WRITE(6,11)NIEN
   11 FORMAT(* ENTER NUMBER OF ENTRY TO REMOVE *,I4,* MAX*)
      CALL RST(6)
      CALL INUM(NCAUSE)
      IF((NCAUSE.GE.1).AND.(NCAUSE.LE.NIEN))GOTO100
   20 WRITE(6,21)NCAUSE
   21 FORMAT(* NUMBER *,I4,* OUT OF RANGE *)
      RETURN
C**********************
  100 WRITE(6,101)NCAUSE
  101 FORMAT(* ENTRY NUMBER *,I5)
      CALL ATOUT(NCAUSE)
      N=IEN(NCAUSE,4)
      WRITE(6,105)(NAME(N,K),K=1,4)
  105 FORMAT(* IN *,4A10)
  110 WRITE(6,111)
```

```
  111 FORMAT(* CORRECT≥ *)
      CALL RST(6)
      READ(5,113) ANS
  113 FORMAT(8A10)
      CALL RST(5)
      IF(ANS.EQ.3HYES)GOTO150
      IF(ANS.EQ.2HNO)GOTO10
      IF(ANS.NE.5H$STOP)GOTO110
C HERE TO STOP, RETURN TO STOPSTA
      RETURN
C*** HERE WITH CORRECT NUMBER
  150 LCAUSE=IEN(NCAUSE,1)
      IF(LCAUSE.LT.LEVEL)GOTO200
C HERE WITH ENTRY ON PRESENT LEVEL, MODIFY DIRECTLY
      CALL ENTEST(NCAUSE,ISTOP)
      RETURN
C HERE MODIFICATION MUST BE TRACED
  200 WRITE(6,201)LCAUSE
  201 FORMAT(* ENTRY AT LEVEL*,I4)
C STACK CAUSE ENTRY
      ISP=1
      ISTK(ISP)=NCAUSE
C START OF MAIN LOOP, TAKE TOP ENTRY ON STACK
C AND GET ALL DEPENDENT ENTRIES STACKED
  220 N=ISTK(ISP)
      CALL STACK(N,NCT)
C IF ANY WERE FOUND, CONTINUE STACKING
      IF(NCT.NE.0)GOTO220
C IF HERE, ENTRY ON STACK TOP HAS NO DEPENDENT ENTRIES
  230 NCOL=IEN(N,3)
C ENTRY MAY HAVE BEEN COLLECTED, IF SO, FOLLOW COLLECTED ENTRY
      IF(NCOL.EQ.0)GOTO250
C HERE, POINTS TO A COLLECTED ENTRY WHICH CAN BE PLACED ON STACK
      ISP=ISP+1
      ISTK(ISP)=NCOL
C RELABLE ANY ENTRIES POINTING TO THIS COLLECTED ONE
      DO 240 J=1,NIEN
      IF(IEN(J,3).NE.NCOL)GOTO240
      IEN(J,3)=0
      IEN(J,7)=1
C STATUS=MUST FOLLOW
  240 CONTINUE
C NOW STACK ENTRIES DUE TO COLLECTED ENTRY
      GOTO220
C*************
C HERE WITH ENTRY WHICH FOR SURE DOES NOT GO ANYWHERE
  250 N=ISTK(ISP)
      DO 260 K=1,7
  260 IEN(N,K)=0
      ISP=ISP-1
```

```
C IS STACK FINALLY EMPTY>
      IF(ISP.GT.0)GOTO220
C HERE AFTER ALL DEPENDENT ENTRIES HAVE BEEN REMOVED OR RELABELED
C INCLUDING NCAUSE NOW COLLAPSE IEN ARRAY AND RECOVER SPACE
C SEARCH FOR ZERO ENTRIES, STARTING WITH NCAUSE
      LOC=NCAUSE-1
  300 LOC=LOC+1
      IF(LOC.GT.NIEN)GOTO400
      IF(IEN(LOC,4).NE.0)GOTO300
C HERE LOC POINTS TO ZERO ENTRY .LE.NIEN HUNT FOR NEXT FILLED ENTRY
      NFIL=LOC
  310 NFIL=NFIL+1
      IF(NFIL.GT.NIEN)GOTO400
      IF(IEN(NFIL,4).EQ.0)GOTO310
C HERE NFIL = NEXT FILLED ENTRY
C WE WILL MOVE NFIL TO LOC, RESETING ALL REFERENCES
C DEPENDING ON WHETHER NFIL IS A COLLECTED ENTRY
      IF(IEN(NFIL,3).NE.0)GOTO330
C HERE NOT A COLLECTED ENTRY
      DO 320 J=1,NIEN
C CHANGE ALL REF FROM NFIL TO LOC
      IF(IEN(J,2).EQ.NFIL)IEN(J,2)=LOC
  320 CONTINUE
      GOTO340
C CHANGE COLLECTED ENTRY REF
  330 DO 336 J=LOC,NIEN
      IF(IEN(J,3).EQ.NFIL)IEN(J,3)=LOC
  336 CONTINUE
C NOW MOVE ENTRY
  340 DO 344 K=1,7
      IEN(LOC,K)=IEN(NFIL,K)
  344 IEN(NFIL,K)=0
      GOTO300
C*************
C HERE AFTER LOC OR NFIL EXCEEDS NIEN, HUNT DOWNWARD FOR NIEN
  400 NIEN=NIEN+1
  410 NIEN=NIEN-1
      IF(NIEN.LE.0)STOP3
C GROSS ERROR
      IF(IEN(NIEN,4).EQ.0)GOTO410
      LEVEL=LCAUSE
      WRITE(6,433)LEVEL,NIEN
  433 FORMAT(* WILL RESTART AT LEVEL=*,I3,* TOTAL ENTRIES=*,I4)
      RETURN
      END
```

```
      SUBROUTINE STACK(KS,NCT)
C KS=ENTRY NUMBER, ROUTINE STACKS ALL ENTRIES WHICH WERE
C GENERATED FROM KS . NCT=COUNT OF ENTRIES STACKED
      COMMON/EXEC/NIEN,NICOM,LEVEL,KSTATUS,ICTRL(10)
      COMMON/STAK/ISP,ISTKMX,ISTK(200)
C STACK POINTER & STACK
      DATA ISTKMX/200 /
      COMMON/ / IEN(400,7),ICOM(400,9),INDEX(400)
C**********************
      NCT=0
      DO 100 J=1,NIEN
      JT=NIEN-J+1
C SEARCH FROM BACK TO FRONT SO STACK WILL BE IN ORDER
      IF(IEN(JT,2).NE.KS)GOTO100
C PLACE ON STACK
      ISP=ISP+1
      IF(ISP.GT.ISTKMX)GOTO200
      NCT=NCT+1
      ISTK(ISP)=JT
  100 CONTINUE
      RETURN
  200 WRITE(6,201)
  201 FORMAT(* STACK SIZE EXCEEDED *)
      STOP
      END
```

6. DETAILED DOCUMENTATION OF THE WRKSHT PROGRAM

GENERAL DISCUSSION

The purpose of the WRKSHT program is to take a case file generated by EVAL and print all of the decisions and comments in a format similar to the worksheet used in the manual ESC method. The case file can be at any stage of completion, thus it is quite convenient to run the WRKSHT program after a given level of analysis has been completed or a substantial number of new entries have been created in order to obtain a summary of progress. WRKSHT does not alter the case file in any way. WRKSHT can be run in batch mode since all output goes to the system printer.

Flow through the main WRKSHT program is extremely simple, with most of the work being done in the PRNT subroutine. After initializing the arrays, subroutine DBIO is called to read in the ESD data base. The NAME array is the only portion of the data base actually used. Next the case file is read in, and the case title is printed.

The IEN array is scanned to locate the last PEA entry (level = 0). The major program loop is now entered and cycled once for each PEA entry. After verifying that JPEA points to a PEA entry, the PAGE subroutine is called to start a fresh page. A call to the PRNT subroutine causes all entries related to this PEA to be printed, ending the loop. After the last PEA entry has been handled, the program stops.

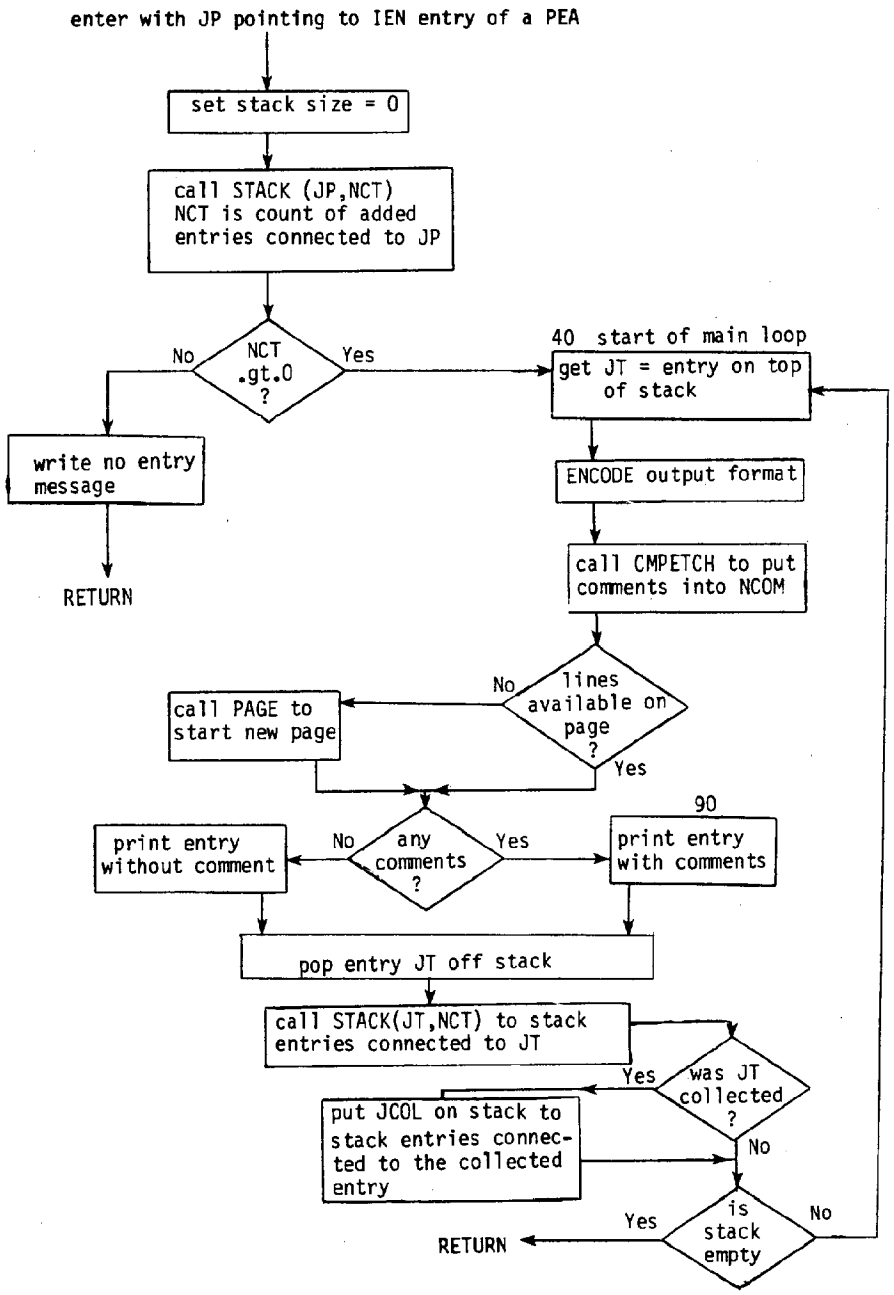
SUBROUTINES USED WITH WRKSHT PROGRAM

SUBROUTINE PRNT(JP)

This routine (Figure 19) is called once for each PEA entry. It handles the entire job of printing the entries dependent on a given PEA, pointed to by JP. On entry, the stack is filled with all entry numbers which are dependent on entry JP. Refer to the data structure discussion for program EVAL and the discussion of the STACK subroutine to see how this is accomplished. Naturally, the entries in the stack at this stage are all at level one.

Figure 19

FLOWCHART FOR SUBROUTINE PRNT



Now the major loop of the subroutine is entered; JT is set to the top entry in the stack. Next, the output format is created according to the level (JLEV) of this entry. This output format creates the indented structure of the final output. Because of the variable format, the comments cannot be printed directly but have to be placed in a single long array NCOM by subroutine CMFETCH.

Actual printing is done by one of two statements depending on whether or not comments must be printed too. (This is done to avoid problems with the implied DO loop when NCMNT=0.)

Now the entry which was just printed is popped off the stack. Then STACK is called to place entries connected to JT on the stack. If entry JT was collected (that is, had no direct evaluation but was collected into another entry on the same level), the collected entry has all of the consequences at the next level. Thus, the collected entry must be placed on the stack and the major loop repeated. This results in the collected entry and all dependent entries being repeated under several lower level entries and possibly several PEAs.

The major cycle is repeated, collecting and popping off entries on the stack until there are no entries left, which indicates that all entries related to this PEA have been printed.

SUBROUTINE STACK(KS,NCT)

This routine is entered with KS pointing to an entry in the IEN array. It searches through all of the entries in the case file for entries which were caused by entry KS, and it places a pointer for each such entry in the ISTK array (the "stack"). The number of entries added is returned as NCT. Note that the IEN array is searched from back to front so that the top of the stack will always be the "first" entry that is due to enter KS.

SUBROUTINE PAGE(JNAM)

This routine simply prints a page header, with the case title and the name of the PEA corresponding to JNAM. The variable ICTRL(2) is used to count the lines used up on this page, while ICTRL(1) counts pages.

SUBROUTINE CMFETCH(JT)

This routine is entered with JT pointing to the first comment attached to a given entry. After checking JT for proper range, it places words from the ICOM array into the NCOM array, stopping when a blank word or the end of an ICOM row is encountered. As can be seen by referring to the General Discussion section of Chapter 5, the linkage of additional comment lines is indicated by a nonzero pointer in ICOM(JT,1). If additional comments are present, they are also inserted into the NCOM array.

SUBROUTINE DBIO(N)

This routine reads the ESD data base. This is the same routine as used in EVAL and SETUP, stripped of unnecessary writing functions.

SUBROUTINE MAGSET(JMAG)

On entering, JMAG is a four-digit integer indicating time, direction, magnitude, and probability judgments. This routine converts it into an A10 format word containing the abbreviations for time, etc.

PROGRAM LISTING

An example of the output from the WRKSHT program is reproduced in this section. At the end of the program listing, part of the case file is shown. The case file output is given to about one-fourth through the third-level attribute changes.

```
      PROGRAM WORKSHT(TAPE2,TAPE3,TAPE6)
C ESD MATRIX FILE, CASE FILE, OUTPUT FILE
C USES MANY ROUTINES FROM EVAL PROGRAM
C CASE FILE AND ESD MATRIX ARE USED TO WRITE A SUMMARY IN
C A FORMAT SIMILAR TO THE AAR WORKSHEET
      COMMON/ESD/NNAME,NPEA,NAME(90,4),ND(90,50),NTITLE(8)
      COMMON/EXEC/NIEN,NICOM,LEVEL,KSTATUS,ICTRL(10)
      COMMON/ / IEN(400,7),ICOM(400,9),INDEX(400)
C ALL VARIABLES HAVE SAME USE AS IN EVAL PROGRAM
C INITIALIZE ARRAYS JUST TO BE ON THE SAFE SIDE
      DO 10 J=1,400
      INDEX(J)=0
      DO 6 K=1,7
    6 IEN(J,K)=0
      DO 10 K=1,9
   10 ICOM(J,K)=0
```

```
C READ FSD MATRIX
      CALL DBIO(1)
C READ CASE FILE
      REWIND 3
      READ(3,31)NIEN,NICOM,LEVEL,KSTATUS,(ICTRL(K),K=1,10)
   31 FORMAT(14I5)
      IF(EOF,3)34,40
   34 WRITE(6,35)
   35 FORMAT(* NO CASE FILE FOUND*)
      STOP
   40 DO 50 J=1,NIEN
   50 READ(3,51)(IEN(J,K),K=1,7)
   51 FORMAT(9I5)
      DO 60 J=1,NICOM
   60 READ(3,61)(ICOM(J,K),K=1,9)
   61 FORMAT(I7,3X,8A10)
      WRITE(6,71)(ICOM(1,K),K=2,9)
   71 FORMAT(*1 TITLE OF CASE FILE *,8A10,//)
      WRITE(6,75)NIEN,NICOM,LEVEL
   75 FORMAT(* ENTRIES=*,I4,* COMMENTS=*,I4,* HIGHEST LEVEL=*,I4)
      IF(KSTATUS.EQ.4)WRITE(6,81)
   81 FORMAT(* CASE STATUS = FINISHED*)
      IF(KSTATUS.NE.4)WRITE(6,83)KSTATUS
   83 FORMAT(* CASE STATUS = *,I3)
C*****************************
C NOW MAJOR LOOP GOES THROUGH THE PEA (LEVEL=0) ENTRIES
C NOTE THAT SINCE ENTRIES ARE ZEROED OUT AFTER BEING PRINTED
C WE HAVE TO LOCATE LEVEL 0 RANGE FIRST
C*****************************
      MXL=1
      DO 110 J=1,NIEN
      IF(IEN(J,1).GT.0)GOTO110
      IF(IEN(J,4).LE.NNAME)GOTO110
      MXL=J
  110 CONTINUE
C ZERO PAGE COUNTER
      ICTRL(1)=0
C MAJOR LOOP
      DO 200 JPEA=1,MXL
      JPNAME=IEN(JPEA,4)
      IF(JPNAME.LT.(91-NPEA))STOP1
      IF(JPNAME.GT.90)STOP2
C SIMPLE ERROR CHECK
      CALL PAGE(JPNAME)
C WRITE PAGE HEADER AND ZERO COUNTES FOR LINES, ETC.
      CALL PRNT(JPEA)
C MAJOR OUTPUT ROUTINE
  200 CONTINUE
      WRITE(6,301)
  301 FORMAT(* FINISHED CASE FILE*)
      STOP
      END
```

```
      SUBROUTINE STACK(KS,NCT)
C KS=ENTRY NUMBER, ROUTINE STACKS ALL ENTRIES WHICH WERE
C GENERATED FROM KS , NCT=COUNT OF ENTRIES STACKED
      COMMON/EXEC/NIEN,NICOM,LEVEL,KSTATUS,ICTRL(10)
      COMMON/STAK/ISP,ISTKMX,ISTK(200)
C STACK POINTER & STACK
      DATA ISTKMX/200 /
      COMMON/ / IEN(400,7),ICOM(400,9),INDEX(400)
C**********************
      NCT=0
      DO 100 J=1,NIEN
      JT=NIEN-J+1
C SEARCH FROM BACK TO FRONT SO STACK WILL BE IN ORDER
      IF(IEN(JT,2).NE.KS)GOTO100
C PLACE ON STACK
      ISP=ISP+1
      IF(ISP.GT.ISTKMX)GOTO200
      NCT=NCT+1
      ISTK(ISP)=JT
  100 CONTINUE
      RETURN
  200 WRITE(6,201)
  201 FORMAT(* STACK SIZE EXCEEDED *)
      STOP
      END
```

```
      SUBROUTINE PAGE(JNAM)
C WRITE PAGE HEADER WITH PEA NAME POINTED TO BY JNAM
C AND FIX LINE & PAGE COUNTERS
      COMMON/EXEC/NIEN,NICOM,LEVEL,KSTATUS,ICTRL(10)
C ICTRL(1)=PAGE COUNT  ICTRL(2)=LINE COUNT
      COMMON/ESD/NNAME,NPEA,NAME(90,4),ND(90,50),NTITLE(4)
      COMMON/ / IEN(400,7),ICOM(400,9),INDEX(400)
C**************
      ICTRL(1)=ICTRL(1)+1
      WRITE(6,5)(ICOM(1,K),K=2,9),ICTRL(1)
    5 FORMAT(*1 CASE FILE= *,8A10,*  PAGE *,I3)
      WRITE(6,11)(NAME(JNAM,K),K=1,4)
   11 FORMAT(* PEA= *,4A10,/)
      ICTRL(2)=3
      RETURN
      END
```

```
      SUBROUTINE CMFETCH(JT)
C TAKE COMMENTS STARTING AT JT AND PLACE INTO NCOM() ARRAY
      COMMON/CMNT/NCMNT,NCMAX,NCOM(100)
      DATA NCMAX/100/
      COMMON/EXEC/NIEN,NICOM,LEVEL,KSTATUS,ICTRL(10)
      COMMON/ / IEN(400,7),ICOM(400,9),INDEX(400)
      INTEGER BLANK
      DATA BLANK/10H          /
C*************************
C COMMENTS ARE PLACED INTO NCOM UNTIL A 10 CHAR BLANK IS REACHED
      NCMNT=0
      IF(JT.EQ.0)RETURN
   20 IF((JT.GT.NICOM).OR.(JT.LT.0)) GOTO80
      DO 40 K=2,9
      IF(ICOM(JT,K).EQ.BLANK)GOTO60
      NCMNT=NCMNT+1
      NCOM(NCMNT)=ICOM(JT,K)
   40 CONTINUE
C CHECK FOR CONTINUED COMMENTS
   60 IF(ICOM(JT,1).EQ.0)RETURN
C RECYCLE WITH CONTINUED COMMENT
      JT=ICOM(JT,1)
      GOTO20
   80 WRITE(6,81)JT
   81 FORMAT(* CMFETCH ERROR, JT=*,I4)
  100 RETURN
      END



      SUBROUTINE PRNT(JP)
C JP POINTS TO PEA ENTR
      COMMON/ESD/NNAME,NPEA,NAME(90,4),ND(90,50),NTITLE(8)
      COMMON/EXEC/NIEN,NICOM,LEVEL,KSTATUS,ICTRL(10)
      COMMON/STAK/ISP,ISTKMX,ISTK(200)
      COMMON/CMNT/NCMNT,NCMAX,NCOM(100)
      COMMON/ / IEN(400,7),ICOM(400,9),INDEX(400)
      INTEGER IVAR(5)
      DATA IVAR(2),IVAR(5)/10HI4,X,A10,X ,10H )          /
C IVAR IS VARIABLE FORMAT
C*********************
C PRINT PAGE HEADER
      JPNAM=IEN(JP,4)
C START STACK OFF WITH LEVEL 1 ATT
      ISP=0
      CALL STACK(JP,NCT)
      LEVEL=1
      IF(NCT.GT.0)GOTO40
      WRITE(6,21)JP
   21 FORMAT(* NO ENTRIES, JP=*,I3)
      RETURN
```

```
C MAJOR LOOP
   40 JT=ISTK(ISP)
      JLEV=IEN(JT,1)
      JCOM=IEN(JT,6)
      JMAG=IEN(JT,5)
      CALL MAGSET(JMAG)
C JMAG IS NOW A10 FORMAT
      JNAM=IEN(JT,4)
      JCOL=IEN(JT,3)
C SET FORMAT BY LEVEL
      ISKIP=1+5*(JLEV-1)
      IREP=8-(JLEV/2)
      I2SKIP=ISKIP+52
      ENCODE(10,51,IVAR(1)) ISKIP
   51 FORMAT(1H(,I2,7HX,4A10,)
      ENCODE(10,53,IVAR(3)) IREP
   53 FORMAT(I2,8HA10,/(  ))
      ENCODE(10,55,IVAR(4)) I2SKIP,IREP
   55 FORMAT(I2,2HX,,I2,4HA10))
C  CHECK/DEBUG
C NOW GET COMMENTS INTO ARRAY NCOM, COUNT = NCMNT
      CALL CMFETCH(JCOM)
C CHECK SIZE OF PRINT VERSUS AVAILABLE SPACE
C ICTRL(1)=PAGE COUNT  ICTRL(2)=LINES THIS PAGE
      ICTRL(2)=ICTRL(2)+1+(NCMNT/IREP)
      IF(ICTRL(2).LT.61)GOTO80
C HERE NEW PAGE IS NEEDED
      CALL PAGE(JPNAM)
      ICTRL(2)=ICTRL(2)+1+(NCMNT/IREP)
   80 IF(NCMNT.GT.0)GOTO90
      WRITE(6,IVAR)(NAME(JNAM,K),K=1,4),JT,JMAG
      GOTO92
   90 WRITE(6,IVAR)(NAME(JNAM,K),K=1,4),JT,JMAG,(NCOM(K),K=1,NCMNT)
C POP OFF PRINTED ENTRY AND ZERO IT OUT
   92 ISP=ISP-1
      DO 94 K=1,7
   94 IEN(JT,K)=0
C AFTER WRITING LANE, STACK ALL DEPENDEN ENTRIES
      CALL STACK(JT,NCT)
C IF NO DIRECT DEPANDENT ENTRIES, POSSIBLE COLLECTED ENTRY
C INDICATED BY JCOL.NE.0
      IF(JCOL.EQ.0)GOTO100
      CALL STACK(JCOL,NCT)
C STACK NOW CONTAINS ALL REFERENCES TO THE CURRENT ENTRY
  100 IF(ISP.LE.0)RETURN
C ONLY RETURN FROM THIS LOOP IS EXHAUSTING STACK
      GOTO40
      END
```

```
      SUBROUTINE DBIO(N)
C N=1 READ,  N=0 WRITE DATA BANK OUT TO TAPE8
      COMMON/ESD/NNAME,NPEA,NAME(90,4),ND(90,50),NTITLE(8)
      REWIND 8
C
    5 FORMAT(8A10,4I4)
   11 FORMAT(4A10,5X,50I1)
   15 FORMAT(* DATA BASE *,/,X,8A10,/* WITH *,I4,* ATT, PEA=*,I4)
C  HERE TO READ
C
  100 READ(8,5) (NTITLE(J),J=1,8),NNAME,NPEA
      IF(NNAME.GT.0) GOTO108
      WRITE(6,103)
  103 FORMAT(* INVALID, NO DATA BASE*)
      STOP1
  108 DO 110 J=1,90
  110 READ(8,11)(NAME(J,K),K=1,4),(ND(J,K),K=1,NNAME)
      WRITE(6,113)
  113 FORMAT(* READ*)
      WRITE(6,15)(NTITLE(K),K=1,8),NNAME,NPEA
      RETURN
      END
C
C



      SUBROUTINE MAGSET(JMAG)
C JMAG INPUT AS 4 DIGIT NUMBERR, OUT AS A10
      COMMON/MAGWDS/IT(3),IM(10),ID(3),IP(4)
      DATA (IT(J),J=1,3)/3H   ,3HLO ,3HSH  /
      DATA (IM(J),J=1,5)/3H   ,3HM9 ,3HM8 ,3HM7 ,3HM6  /
      DATA (IM(J),J=6,10)/3HM5 ,3HM4 ,3HM3 ,3HM2 ,3HM1  /
      DATA (ID(J),J=1,3)/3H   ,3HI  ,3HD   /
      DATA (IP(J),J=1,4)/3H    ,3HDF ,3HPR ,3HP0  /
C SEPARATE INDEXES
      J3=JMAG/1000
      J2=(JMAG-(J3*1000))/100
      J1=(JMAG-(J3*1000)-(J2*100))/10
      J0=JMAG-(J3*1000)-(J2*100)-(J1*10)
C TEMP CHECK
      ENCODE(10,15,JMAG)IT(J3+1),IM(J2+1),ID(J1+1),IP(J0+1)
   15 FORMAT(A3,A3,A2,A2)
      RETURN
      END
```

CASE FILE= PILOT STUDY
PEA= CONSUMER REMOVAL

BENTHIC COMMUNITY	11	SH	M7	D	OF	COMPLETE REMOVAL,438AC,E/E=100+ INSTANTANEOUS.		
SEDIMENT NUTRIENTS	37	SH	M5	D	PR	EQUILIBRIUM RAPIDLY RE-ESTABLISHED		
SEDIMENT DISSOLVED OXYGEN	38	SH	M1		PR	OTHER EFFECTS MORE IMPORTANT		
SUSPENDED ORGANIC MATTER/BACTERIA	39	SH	M5	I	PO	HERBIVORES,DETRITIVORES ARE PROBABLY THE MAJOR CONSUMERS OF ORG. M. IN THE BAY. EFFECT OF BENTHICS SMALL		
SEDIMENT ORGANIC MATTER/BACTERIA	40	SH	M5	D	PR	THIS IS CONTRIBUTION OF ORG. M. BY BENTHICS DUE TO FILTER FEEDING AND SUBSEQUENT EXCRETION		
PHYTOPLANKTON	41	SH	M4	I	PO	SIMILAR REASONING AS WITH SUSP. ORG. M.		
INTERMEDIATE CONSUMERS	42	SH	M6	D	PR	ASSUME 5000 ACRE RANGE, 4.4 PCT DECREASE		
DISSOLVED AND SUSPENDED NUTRIENTS	109	SH	M1		PR	OTHER NUTRIENT SOURCES ARE MUCH MORE SIGNIFICANT		
DISSOLVED OXYGEN	110	SH	M4	I	PO	NEGLIGABLE IN RELATION TO OTHER SOURCES AND SINKS OF OXYGEN		
SUSPENDED ORGANIC MATTER/BACTERIA	111							
HERBIVORES AND DETRITIVORES	112							
BENTHIC COMMUNITY	113							
MIGRATING INTERMEDIATE CONSUMERS	114							
TOP CONSUMERS	115							
BIRDS	116							
TOP CONSUMERS	43	SH	M6	D	PR	DECREASE IN 0.9 PCT OVER 16000 ACRES		
DISSOLVED AND SUSPENDED NUTRIENTS	117	SH	M1		PR			
DISSOLVED OXYGEN	118	SH	M1		PR	TOP CONSUMERS ARE ONLY A MINOR OXYGEN CONSUMER		
SUSPENDED ORGANIC MATTER/BACTERIA	119							
HERBIVORES AND DETRITIVORES	120							
BENTHIC COMMUNITY	121							
INTERMEDIATE CONSUMERS	122							
MIGRATING TOP CONSUMERS	123							
BENTHIC COMMUNITY	12	LO	M4	D	PR			
SEDIMENT NUTRIENTS	44	LO	M4	D	PO			
SEDIMENT DISSOLVED OXYGEN	45	LO	M1		PR	WATER DO IS MUCH MORE IMPORTANT		
SUSPENDED ORGANIC MATTER/BACTERIA	46	LO	M4	I	PO	SEE COMMENT FOR SHORT		
SEDIMENT ORGANIC MATTER/BACTERIA	47	LO	M5	D	PR	SEE SHORT COMMENTS		
PHYTOPLANKTON	48	LO	M3	I	PO			
INTERMEDIATE CONSUMERS	49	LO	M6	D	PR	ASSUME 5000 ACRE RANGE, 0.88 PCT DELTA E LOSS		
DISSOLVED AND SUSPENDED NUTRIENTS	124	LO	M4	D	PR			
DISSOLVED OXYGEN	125	LO	M1		PR			
SUSPENDED ORGANIC MATTER/BACTERIA	126							
HERBIVORES AND DETRITIVORES	127							
BENTHIC COMMUNITY	128							
MIGRATING INTERMEDIATE CONSUMERS	129							
TOP CONSUMERS	130							
BIRDS	131							
TOP CONSUMERS	50	LO	M6	D	PR	EST AT 0.1 PCT OVER 16000 ACRES - HIGHLY MOBILE TOP CONSUMERS ARE ASSUMED TO BE ABLE TO SHIFT FOOD ITEMS		
DISSOLVED AND SUSPENDED NUTRIENTS	132	LO	M4	D	PR			
DISSOLVED OXYGEN	133	LO	M1		PR			
SUSPENDED ORGANIC MATTER/BACTERIA	134							
HERBIVORES AND DETRITIVORES	135							
BENTHIC COMMUNITY	136							
INTERMEDIATE CONSUMERS	137							
MIGRATING TOP CONSUMERS	138							

```
SUSPENDED SOLIDS                    13 SH M7 I DF
     HEAT IN WATER COLUMN                51 SH M2   DF EFFECT INSIGNIFICANT DUE TO STRONG MIXING
     IMPORT/EXPORT INORGANIC SOLIDS      52 SH M4 I PR EFFECTS CONFINED TO OPEN BAY
     SEDIMENT SOLIDS                     53 LO M2 I PO ACCUMULATION NEAR CUTTER-HEAD IS SMALL IN EXTENT
     PHYTOPLANKTON                       54 SH M5 D PR EFFECT ON THE ORDER OF 100 PCT OVER 8 ACRES AT ANY ONE TIME DURING PRO
                                                      JECT
     HERBIVORES AND DETRITIVORES         55 SH M5 D PR MOBILE ORGANISMS CAN AVOID HIGH CONCENTRATION OF SUSP. SOLIDS
                                                      EFFECT ON EGGS AND LARVAE NOT CONSIDERED, SEE ESPEY HUSTON REPORT
                                                      PP B-58 TO B-61
     BENTHIC COMMUNITY                   56 SH M6 D PR THIS IS DECREASE OFF SITE DUE TO SEDIMENTATION , AREA ABOUT 32 ACRES
                                                      WITH RECOVERY ON ORDER OF 80 PCT WITHIN 2 YRS
          SEDIMENT NUTRIENTS                  139 SH M4 D PR
          SEDIMENT DISSOLVED OXYGEN           140
          SUSPENDED ORGANIC MATTER/BACTERIA   141
          SEDIMENT ORGANIC MATTER/BACTERIA    142
          PHYTOPLANKTON                       143
          INTERMEDIATE CONSUMERS              144
          TOP CONSUMERS                       145
     INTERMEDIATE CONSUMERS              57 SH M5 D PR MOBILE ORGANISMS CAN AVOID PROBLEM, SEE HERBIVORE COMMENTS
     TOP CONSUMERS                       58 SH M5 D PR MOBILE ORG CAN AVOID AREA OF CONCENTRATION, LITTLE EFFECT
SUSPENDED SOLIDS                    14 LO M1   DF
DISSOLVED AND SUSPENDED NUTRIENTS   15 SH M6 I DF FPC,V,2,1977,PP,221/222
     IMPORT/EXPORT NUTRIENTS             59 SH M2 I PO EFFECTS CONFINED TO AREA NEAR SITE
     SEDIMENT NUTRIENTS                  60 SH M4 I PR EFFECT SMALL RELATIVE TO SEDIMENT RESERVOIR
     PHYTOPLANKTON                       61 SH M5 I PR STIMULATION PROBABLY OVER LARGER AREA THAN SHADING EFFECT
DISSOLVED AND SUSPENDED NUTRIENTS   16 LO M1   DF NUTRIENTS DISPERSED OR FIXED SOON AFTER RELEASE.
```

COASTAL ZONE
INFORMATION CENTER

DATE DUE

GAYLORD No. 2333			PRINTED IN U.S.A.

NOAA COASTAL SERVICES CENTER LIBRARY
3 6668 14106 7142